AD-A117 091 SOFTWARE ARCHITECTURE AND ENGINEERING INC ARLINGTON VA F/G 9/2
SOFTWARE ACQUISITION MANAGER'S KNOWLEDGE-BASED EXPERT SYSTEM.(U)
JUN 82 A B FERRENTINO N00014-82-C-0130
UNCLASSIFIED NL

(12)

Report N00014-82-C-0130

SOFTWARE ACQUISITION MANAGER'S KNOWLEDGE-BASED EXPERT SYSTEM

Summary of Phase 1 Feasibility Analysis

Andrew B. Ferrentino

Software Architecture and Engineering, Inc.
Suite 1220
1401 Wilson Boulevard
Arlington, Virginia 22209

30 June 1982

Final Report for Period January 1 to June 30, 1982

Prepared for:

Department of the Navy
Office of Naval Research
Arlington, Virginia 22217

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER N00014-82-C-0130	2. GOVT ACCESSION NO. AD-A117091	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Software Acquisition Manager's Knowledge-Based Expert System		5. TYPE OF REPORT & PERIOD COVERED Research Contract Final Report 1/1-6/30/82 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) ANDREW B. FERRENTINO		8. CONTRACT OR GRANT NUMBER(s) N00014-82-C-0130
9. PERFORMING ORGANIZATION NAME AND ADDRESS Software Architecture&Engineering, Inc. Arlington, Virginia		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Virginia		12. REPORT DATE June 30, 1982 13. NUMBER OF PAGES 141
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software acquisition, expert systems, management workstations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(See insert)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

MIL-STD-847A
31 January 1973

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

This report is the final report for work performed under contract to the Office of Naval Research. The contract was sponsored under the Defense Small Business Advanced Technology (DESAT) Program, phase 1. The purpose of the contract was to demonstrate the feasibility of developing a microcomputer-based manager's workstation encorporating expert system technology that can support the DoD acquisition manager in the complex activity of software acquisition for complex weapon systems.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Summary

A serious problem impacting the ability of the Navy to develop and deploy modern weapon systems is the reliability of software and the long lead time to develop software embedded in the weapon systems. The problem is so severe that fleet readiness is jeopardized unless some significant improvements are implemented for the acquisition and development of software. Acquisition management is a key contributing factor to the current problem. There are not enough experienced managers available to handle the complex workload of software acquisition for Navy weapon systems.

A promising technology with the potential of significantly improving the software acquisition productivity and quality is expert systems. The technology of expert systems provides the means of capturing and putting to use expert software acquisition knowledge covering the entire software life cycle. It represents an efficient and productive method of communicating this knowledge to inexperienced acquisition managers and will significantly improve the productivity of experienced acquisition managers. The result to the Navy will be a significant

reduction in cost, an improvement in utility, and a substantial increase in the reliability of software developed for mission critical systems.

The software acquisition manager's workstation (SAM/WS) will be a microcomputer-based, interactive system designed with an easily understood user interface. It will offer expert support on seventeen subjects including the acquisition process, requirements, request for proposal preparation, and monitoring of contractor progress and costs. It also will support the generation of reports and documentation, aid in resource estimation, and support various analyses such as precedence network analysis of schedule dependencies.

The objective of the Defense Small Business Advanced Technology (DESAT) Phase I effort was to demonstrate the feasibility of the SAM/WS. Feasibility issues were identified relating to the user interface, scope of the domain knowledge, software design, workstation characteristics, and magnitude of the development effort. The issues were addressed on the basis of a survey of the state-of-the-art in knowledge-based expert systems, generation of a demonstration requirements (i.e., standards) expert system, and analysis involving top level design of the SAM/WS.

As a result of the DESAT Phase I study, it was concluded that it is feasible to implement a SAM/WS that will

significantly contribute to the solution for reducing Navy software acquisition cost and achieving improved software reliability. This conclusion is based upon the following results derived from this feasibility analysis.

- A user interface to the SAM/WS was defined which will efficiently support a software acquisition manager with little or no computer experience. In addition, the user interface is defined to accommodate the same efficiency and quality of products for acquisition managers ranging in skill level from novice to very experienced.
- A design technique was identified for partitioning the extensive domain knowledge into independent knowledge bases which can interact through a common set of attributes. In addition, this design for partitioning the software facilitates and supports an incremental development of the SAM/WS; thereby, allowing this knowledge-based expert system to be developed in phased evolution, ranging from most essential features to a full capability system.
- A very viable software implementation language was identified that will ensure that the SAM/WS is portable to a variety of hardware. The eventual availability of the Ada language is the

only dependence upon which the strategy is based.

- SAM/WS hardware and support software requirements were identified and there are several micro-computer candidates which will fulfill SAM/WS developmental requirements.
- A development strategy was prepared that will result in the early availability of an operationally useful subset (e.g., most essential near-term capabilities) of the SAM/WS that can be evolved over a three-year period into a full capability SAM/WS.

Given the completeness of the feasibility analysis for implementing the SAM/WS and the potential impact of its use on solving the Navy and DoD software problems, it is recommended that the development strategy summarized below be implemented.

- Fund an initial operationally useful subset of the SAM/WS (Release 1) which supports most essential near-term capabilities such as software acquisition tutorial, HELP, Navy standards, SOW and deliverable preparation.
  (Initial SAM/WS (Release 1) effort is 55.9 man-months and is within the scope of the DESAT Phase II program.)
- Fund the 206.2 man-months for the Full System Additional Effort over a two or three year period

after completion of the initial SAM/WS (Release 1). (Design of SAM/WS facilitates evolution to a full capability through incremental funding for the 206.2 man-months that are required for completion.)

Preface

This is the final report prepared by Software Architecture and Engineering, Inc. (Software A&E) and its subcontractor, Knowledge Engineering, Inc., under the Office of Naval Research (ONR) contract N00014-82-C-0130. The research was sponsored under Phase I of the Defense Small Business Advanced Technology (DESAT) Program. This report describes the analysis performed to evaluate the feasibility of developing an expert system workstation designed to support Navy software acquisition managers.

The principal investigator for the Phase I effort was Mr. Andrew Ferrentino. He was supported by Mr. Robert Whaley, also of Software A&E. Valuable expertise in the design and implementation of expert systems was provided by Dr. James Reggia and Mr. Barry Perricone of Knowledge Engineering, Inc. Mr. Charles Hager of JCL Associates provided consultation in the development of a software standards demonstration knowledge-based expert system. The Knowledge Engineering, Inc. expert system product, Knowledge Engineering System (KES), was used to generate the standards demonstration.

Table of Contents

1. INTRODUCTION

This report documents the results of the Phase 1 efforts of Software Architecture and Engineering, Inc. (Software A&E) and Knowledge Engineering, Inc., a subcontractor, under the Defense Small Busines Advanced Technology (DESAT) Program. The objective of the Phase 1 effort was to show the feasibility of developing a microcomputer-based workstation hosting an expert system for the support of Navy software acquisition managers. Such a workstation would assist an inexperienced manager to perform adequately in this complex role while avoiding the many pitfalls that lead to costly overruns or unnecessarily large life cycle costs. It also would be a useful aid to the experienced manager, improving the quality and completeness of his product.

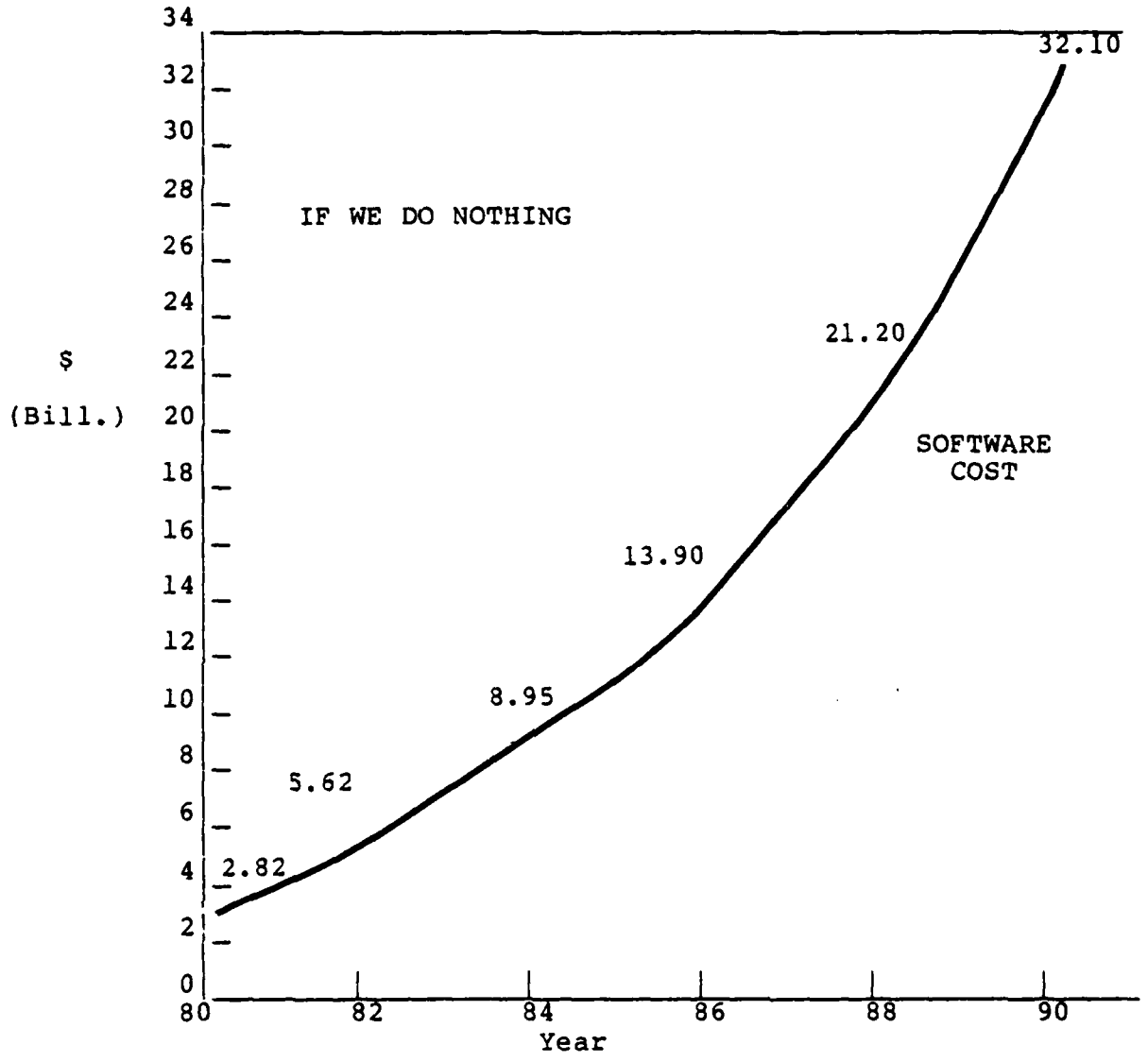
The body of the report identifies the key feasibility issues and describes the work performed in analyzing these issues. Sections 3 and 4 provide the conclusions and recommendations resulting from the feasibility analyses. The remainder of the Introduction provides background information which motivates the development of a software acquisition managers workstation (SAM/WS). It describes the capabilities of the SAM/WS, and it provides an outline of the approach taken in the feasibility analysis.

1.1 Background

Software is becoming a critical factor in the struggle to maintain technological superiority over our adversaries. Modern weapon systems exhibit a trend of increasing dependence on software-intensive embedded computer systems. Although the cost-performance of hardware has dramatically improved over the past 20 years, there has not been a corresponding improvement in software. Cost overruns, schedule delays, and subquality products are the rule rather than the exception in the management of software acquisitions.

This problem is brought into focus when one considers the complexity of software systems and the scope of knowledge required of management. This required knowledge includes Department of Defense (DOD) acquisition policy and documentation standards, the software life cycle process, software engineering techniques, development planning, and the technical aspects of the software system. There are few managers today who possess this requisite knowledge.

One can only speculate on the cost to the Navy due to unqualified software acquisition managers, but all indications are that the stakes are high. As shown in Figure 1, DOD software expenditures in 1982 are estimated at $5.62 billion. Unless software production and quality are dra-



Source: Electronic Industries Association.

Figure 1: DoD Embedded Computer Software Costs

matically improved, the cost of software will grow to an estimated $32 billion in 1990.

The potential management impact on these huge software expenditures can be illustrated through the following two examples:

a. A Navy contractor developed a major tactical embedded computer software system using a development support system hosted on a large commercial computer. After delivery, the Navy found it could not maintain the software without the commercially-hosted support system, but, the latter was not a deliverable under the contract. Therefore, the Navy had to expend additional millions of dollars to procure an adequate support system. This situation should have been diagnosed by the acquisition manager before approving the original development contract.

b. The acquisition package for a major Navy weapon system with an embedded computer did not require documentation for application software which was developed as part of the weapon system. Because there was no documented baseline, configuration control of the application software was chaotic which resulted in minimal fleet readiness for this weapon system. To solve this problem, documentation for the application software was developed by reverse engineering and multiple configuration audits were required to ensure validity of the documentation. An experienced software acquisition manager would have required software

documentation and enforced configuration management, thereby preventing a reduction in fleet readiness and the additional million dollar expenditure for developing the reverse engineering documentation.

The potential savings to the Navy over the next 5 years might be in the billions of dollars if fully qualified management could be assigned to all software acquisitions. Given the severe shortage in qualified management this is an impractical goal unless tools and techniques can be found to magnify the skills of the available personnel. One approach to this would be to capture the composite knowledge of highly qualified software acquisition managers in such a way that it could be used both by less experienced managers to adequately perform in an acquisition capacity and by qualified managers to increase their efficiency.

Two recent trends in technology provide a basis for the capture of requisite knowledge in a form useable by acquisition managers. First, knowledge-based expert systems have been developed and demonstrated effective in such diverse areas as medicine, chemistry, biology, electronics, and geological survey applications. These systems capture basic decision-making information as a knowledge base (i.e., associative knowledge) and a "reasoning" capability which results in performance matching that of an expert (see Appendix A for a state-of-the-art survey).

The second technology trend is the emergence of microcomputer-based personal workstations to support professionals. The primary characteristic of these workstations is that they require no knowledge of programming to use them effectively. The technology associated with each of these trends is discussed briefly below.

An expert system is a system designed to support user-oriented interactions aimed at obtaining information in a specific knowledge area (domain) and/or aiding a decision-making process. Unlike a data base system, an expert system can supply information that is not explicitly available in its information base which can be inferred from the explicitly stored information. An expert system, if properly implemented, can emulate a human expert in some knowledge area. It captures not only the raw knowledge of the expert, but also the inference methods used by the expert in reasoning about problems in his/her domain.

An expert system can be thought of as a system consisting of three components:

a. User Interface - an interactive interface that provides for a natural, friendly dialogue with a user who may be competent in a given knowledge area, but does not necessarily have specialized training or familiarity with computers.

b. Knowledge Base - the information representing an

area of expertise or knowledge domain (e.g., laser physics). This information is represented in a form which can be processed by the inference mechanism(s) of the expert system.

c. Inference Mechanisms - algorithms for inferring information from the explicit data in the knowledge base. This inferred information can be used to formulate expert recommendations or to provide information useful to a decision process.

A personal workstation is an interactive hardware/ software system configured to support an individual in performing a number of tasks fundamental to his/her job assignment. It can be designed to operate in stand-alone mode or as a remote intelligent terminal to a mainframe computer. The workstation hardware and associated software provide a user-friendly interactive interface to various software functions characterizing the workstation application.

The objective of the SAM/WS research project is to merge expert system and personal workstation technologies to provide expert assistance to software system acquisition managers. The resulting workstation will not replace the acquisition manager, rather it will improve the quality of the manager's decisions and assure that major issues are properly addressed and managed.

A secondary objective of the SAM/WS research project

is to develop the software acquisition manager's workstation in a manner that permits evolution of a prototype domain-independent programming environment for creating knowledge-based expert systems. Domain-independence is a characteristic of the expert system software that allows implementation of a new expert system by developing the knowledge base only. No programming is required. Successful attainment of this objective will result in the means to efficiently build expert systems for other applications.

The benefits to the Navy and DOD that can result from the SAM/WS research project are many. It will provide an immediate relief to the problem of an insufficient number of qualified management resources by augmenting the experience of less experienced managers. In addition the productivity of experienced acquisition managers will be increased. This in turn should have a significant impact on the cost and quality of software systems developed by the Navy.

A less obvious benefit of the acquisition manager's workstation will be that it can become the repository of "corporate knowledge" for Navy software acquisition. As experts and non-experts use the system, their experience can be captured in the knowledge base. Over time, acquisition managers may come and go, but the SAM/WS knowledge base will continue to evolve, retaining the experience of

departed experts. This would be a far cry from the situation today where organizations are hamstrung by the departure of a key manager whose knowledge and experience departs with him or her.

Another benefit that will derive from the proposed research effort is the potential cost-effective application of the domain-independent technology to other Navy and DOD problems. These might include command and control, system verification and validation, logistics management, ADP security, and shipboard damage control as well as maintenance of complex equipment. Another obvious area would be medical diagnosis to alleviate the severe shortage of physicians in the Navy.

1.2 SAM/WS Description

The SAM/WS will be an interactive microcomputer-based hardware/software system designed to support a single interactive user. The hardware configuration for the workstation is depicted in Figure 2. The microcomputer will be configured with a disk for mass storage, a document quality printer, and a keyboard CRT terminal. The printer will provide the means of hardcopy reports, and the disk will be used to store knowledge bases, software that drives the workstation, and information pertaining to the acquisition problem which is dynamically generated as a result of user (acquisition manager) interaction.

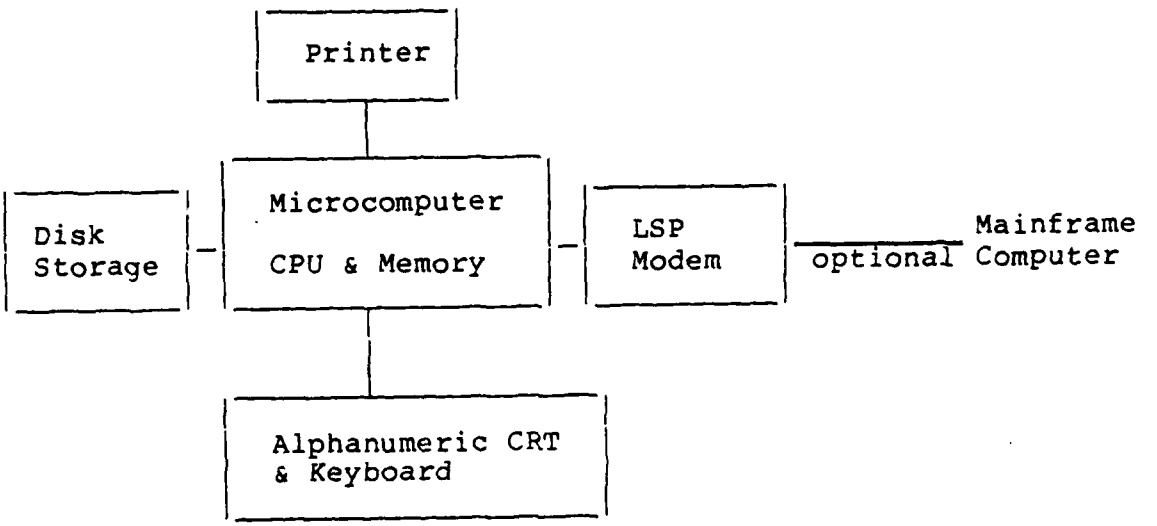


Figure 2: SAM/WS Hardware Configuration

The workstation will be capable of accessing a data base resident on a remote computer. This optional feature will allow access to the data base of a software engineering environment. This data base will contain project measurements and other data that is required by the acquisition manager to monitor and control a development activity. To implement this feature, minimal workstation interface software may have to be included in the software engineering environment.

The acquisition manager's interactive interface to the workstation will not require computer experience for effective operation. It will provide a combination of menu-driven, fill-in-the-blanks, and simple editing protocols for interaction. An intelligent HELP capability and acquisition management tutorials will be provided to ensure that interaction is self-explanatory, even for the novice user.

The primary requirement of the SAM/WS is that it support all responsibilities of the Navy software acquisition manager from Program Objectives Memorandum (POM) to in-service engineering support of software. Figure 3 shows the various activities involved arranged in a precedence hierarchy. Within the various activities shown in Figure 3, the SAM/WS will support the SAM in one or more of the following generic areas:

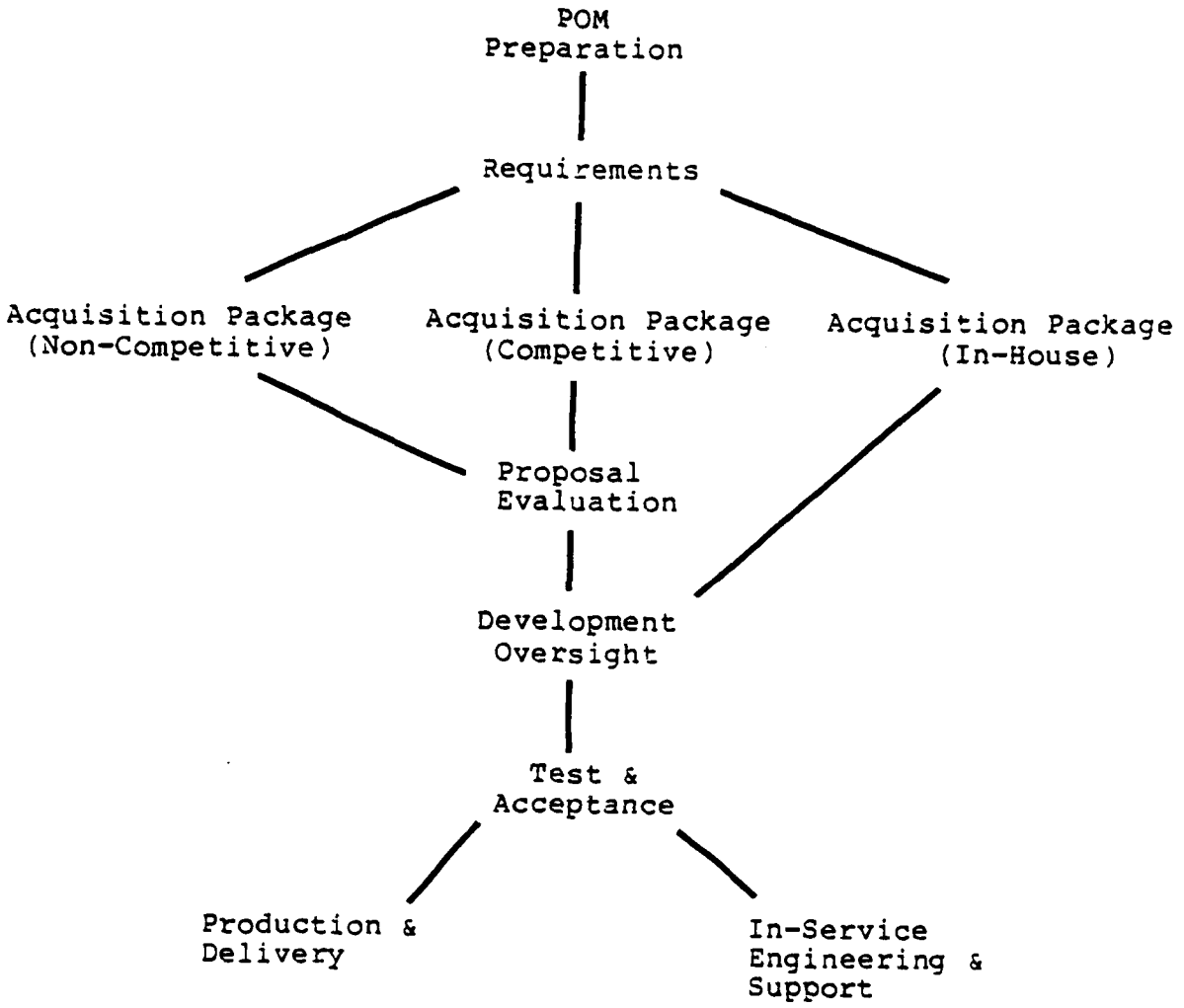


Figure 3: Scope of Software Acquisition Management Support

a. Software Acquisition Guidance: The SAM/WS shall provide tutorial guidance to the acquisition manager for any of the software acquisition activities. It shall support him in the development of a plan and checklist of responsibilities for managing the software acquisition process.

b. Software Acquisition Products: The SAM/WS shall support the acquisition manager in the generation of necessary acquisition documents such as Mission Element Need Statement (MENS), Navy Decision Coordinating Paper (NDCP), Master Information Paper Summary (MINIMIPS), Acquisition Strategy, POM, Statement of Work (SOW), Deliverables (Form 1423), Request for Proposal (RFP), and Navy Standards (GFM and GFE). This shall be accomplished through expert systems which help the acquisition manager tailor an outline and insert "canned" text, and through text editing services to complete the documents.

c. Analysis: The SAM/WS shall support analysis required by the acquisition manager to perform his job properly. These shall include development resource estimation, life cycle cost analysis, and schedule dependencies and critical path analyses.

d. Monitor and Control: The SAM/WS shall support the acquisition manager in monitoring the development activity of the development agent, and in monitoring the Test and Acceptance activity. Services shall include cost

work breakdown structure tracking, milestone tracking, contingency analysis, and acceptance guidance. Analogous activities will carryover into in-service engineering.

To provide the support of the acquisition manager outlined above, the SAM/WS software will provide the following functional capabilities:

a. Interactive Interface: management of the dialogue with the acquisition manager and screen formatting,

b. Expert Systems Components: provision of inference mechanisms and expert system dialogue management,

c. Analyses Tools: resource estimation, scheduling and other acquisition and management tools,

d. Text Editor: text entry and editing,

e. Report Generation: hardcopy report formatting and printing,

f. Data Base Management: storage, retrieval and access management of the SAM/WS data base,

g. Operating System: foreground/background processing and I/O device access,

h. Utilities: knowledge base creation aids and standard system utilities.

i. Knowledge Bases: up to seventeen knowledge bases supporting acquisition planning, RFP preparation, monitoring and controlling, etc.

The SAM/WS software will be designed to maximize flexibility to change through information hiding for

device independence, the use of high level languages, and table-driven software. The expert systems software will be designed to be knowledge base independent (domain independent) thus making it adaptable to other applications.

1.3 Feasibility Issues

There are five technical issues which must be addressed in showing the feasibility of a SAM/WS. These issues are summarized below:

a. User Interface: In most expert systems today the user interfaces have been question and answer type interactive interfaces. Although this mode of interaction will be valuable to the SAM/WS, a diversity of interaction methods will be required. This will require the development of an interface that exhibits the necessary diversity but which remains simple enough for non-programmers to use. The question to be addressed is whether an adequate interface can be developed.

b. The Scope of the Knowledge Base: The domain knowledge associated with software acquisition is very large. Therefore the potential scope of a knowledge base capturing aspects of this domain knowledge is also very large. This raises questions of how long it will take to implement such a knowledge base. The ability to partition the knowledge base such that it can be developed in operationally useful increments would make early availability of

a SAM/WS feasible.

Another issue relating to the knowledge base is the need to interface the knowledge base attribute values with a problem-oriented dynamic data base. The data base will vary with each acquisition addressed and will represent the dynamic generation of information by the user, but the knowledge base would change only when new domain knowledge concerning the acquisition process is to be incorporated. This combination of knowledge base and data base in one system is an extension to expert systems as they have been implemented to-date.

c. Software Design: The SAM/WS software will utilize the approaches used for expert system implementation and more traditional techniques of management analysis. This raises a question as to whether these approaches can be combined effectively. It also raises a question as to the programming language to be used. The LISP programming language is suited to the expert system functions but a language like PASCAL might be better suited to the analysis components.

d. Workstation Characteristics: Once the size and complexity of the knowledge base and the nature of the software architecture is understood, it must be determined if it is feasible to host such a system on a microcomputer given the commercially available microcomputer technology.

e. Development: The software and knowledge bases

required to implement the SAM/WS are substantial. A minimum of twenty man-years over a three year period will be required. Given the magnitude of effort, a strategy to incrementally develop the SAM/WS is needed. Such a strategy must yield operationally useful subsets of the SAM/WS for early evaluation.

1.4 Approach to Feasibility Analysis

The feasibility issues identified in Section 1.3 were addressed through a combination of limited prototyping, analysis, and literature extraction. The limited prototype consisted of a Navy software standards demonstration expert system. This demonstration expert system was created with the aid of Knowledge Engineering System (KES). KES is a proprietary product of Knowledge Engineering Incorporated which supports a domain expert in constructing a knowledge base, and which supplies all the software necessary to become an operational expert system when combined with the knowledge base (see Appendix C for a description of KES).

The demonstration expert system was a source of information for the analysis of:

a. Knowledge base sizing and the feasibility of incremental development by knowledge base partitions.

b. The feasibility of domain independence.

c. User interface requirements.

d. Software sizing estimates and the related requirements on the microcomputer hardware.

The other source of data for analysis of the feasibility issues was the literature on expert systems and the combined experience of the Software A&E research team. These sources were used to outline the SAM/WS knowledge base and software architecture, to define and analyze the user interface, and to define and size the SAM/WS microcomputer system.

2. FEASIBILITY ANALYSIS

The Navy standards knowledge base demonstration and the study analyses of the SAM/WS feasibility issues provide ample evidence that a workstation-based expert system to support a Navy software acquisition manager is attainable. The following subsections document the results of the analyses of the various feasibility issues.

The Navy standards knowledge base demonstration was successful and proved very valuable to the analysis. It provided the means for estimating the size of the SAM/WS knowledge bases, provided improvements needed in the user interface, and identified functions required for the SAM/WS. This information is reflected in the study results in the following subsections. Examples of the Navy standards expert system interaction and a source listing of the demonstration knowledge base are presented in Appendix B.

2.1 User Interface

Most expert systems in operation today support a somewhat limited interactive interface consisting of question and answer as the means of interaction, with a limited command set to manipulate the knowledge base. KES is an example of a well engineered interface of this type that is very user friendly and requires minimal training.

Use of KES in the Navy standards demonstration and projection of the other KES functional capabilities to be provided to the user show that the typical expert system user interface is not sufficient for the SAM/WS. It will be necessary to support other interaction methods such as text entry and editing, graphics, and menus as well as the KES-like interface.

To provide a framework for the SAM/WS interface discussion, a brief summary of the SAM/WS user requirements are presented. This is followed by a discussion of user interface design alternatives that are feasible approaches to meeting the requirements.

2.1.1 SAM/WS User Requirements

Given the scope of SAM/WS support outlined in Section 1.2, it is possible to postulate some of the important requirements for the SAM/WS user interface. These interface requirements can be classified as console modes and interactive characteristics. The console modes to be supported are keyboard entry, cursor or function key selection, alphanumeric and line graphics CRT output, and document quality printed output. The interface characteristics shall include the following:

a. Multiple Screen Segments: The CRT screen will be segmentable to simultaneously display information of different types.

b. Simple Protocols: The method of interacting will be simple, requiring little training to use the basic SAM/WS services. For more experienced users, shorthand inputs will be allowed for efficient interaction.

c. Tutorial: Sufficient tutorial on the acquisition process and use of the SAM/WS will be available to support in-line training and HELP.

d. Error Messages: Meaningful English phrase error messages will be generated for the user when any abnormal situation is encountered.

e. Checkpointing: It will be possible for the user to suspend interaction with a SAM/WS service such that the interaction can be resumed at a later date.

2.1.2 SAM/WS User Interface Design

The primary mode of operation of the SAM/WS will be through interactive sessions at the CRT Terminal. The functional capabilities of the SAM/WS will be grouped into services which can be invoked by the user. Any service of the SAM/WS (e.g., expert system, analysis tool, report generator) will be accessed through a request to a Service Selection Manager. Once a service is selected, the user will interact with it directly. Thus the SAM/WS interactive interface can be envisioned as consisting of two levels:

a. Level 1 - Service Selection: At this level, inter-

action will be accomplished through menu selection from menus ordered in a tree structure of limited depth. This very simple interface will ensure that a person with no training can at least select tutorial services without any complication;

b. Level 2 - Service Interaction: Each service will interact with the user in a manner best suited to the work to be performed. The methods used will include text entry and editing, menu selection, and command language. The syntax for each interaction method will be common across services to facilitate ease of learning and use.

There are several approaches that can be taken to implement multiple screen segments. One approach would be to use a dynamic "windowing" scheme wherein the user can activate as many segments or windows as desired. These windows would correspond to services simultaneously in use by the user as shown in Figure 4.

Another approach would be to produce a screen with a fixed segment format as shown in Figure 5. One segment would be dedicated to error and status messages generated by the SAM/WS. Another segment would be a menu of universal commands available during interaction with any service (e.g., suspend operation and HELP). The main segment would be the service display area. It would be possible to split this area into two segments for the simultaneous interaction with two services (e.g., HELP and the resource

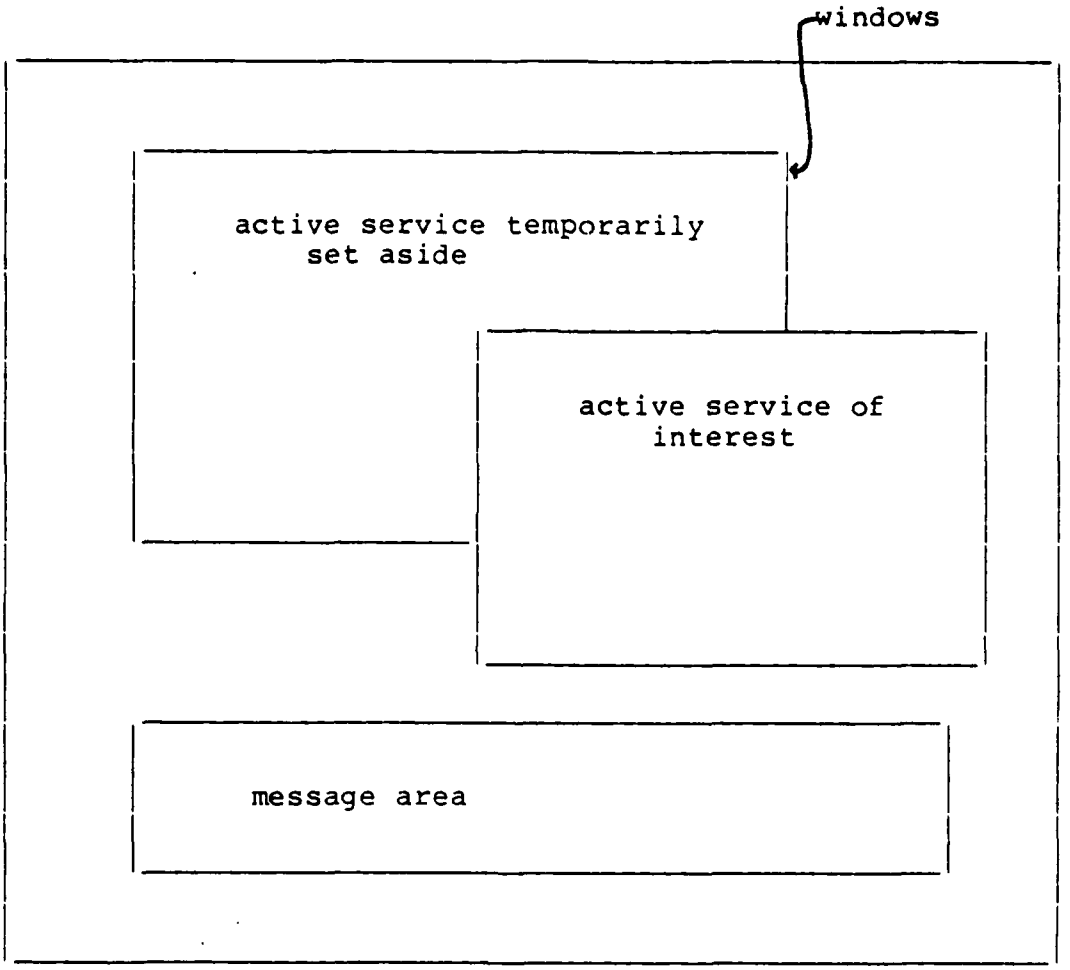


Figure 4: Dynamic Windowing Approach

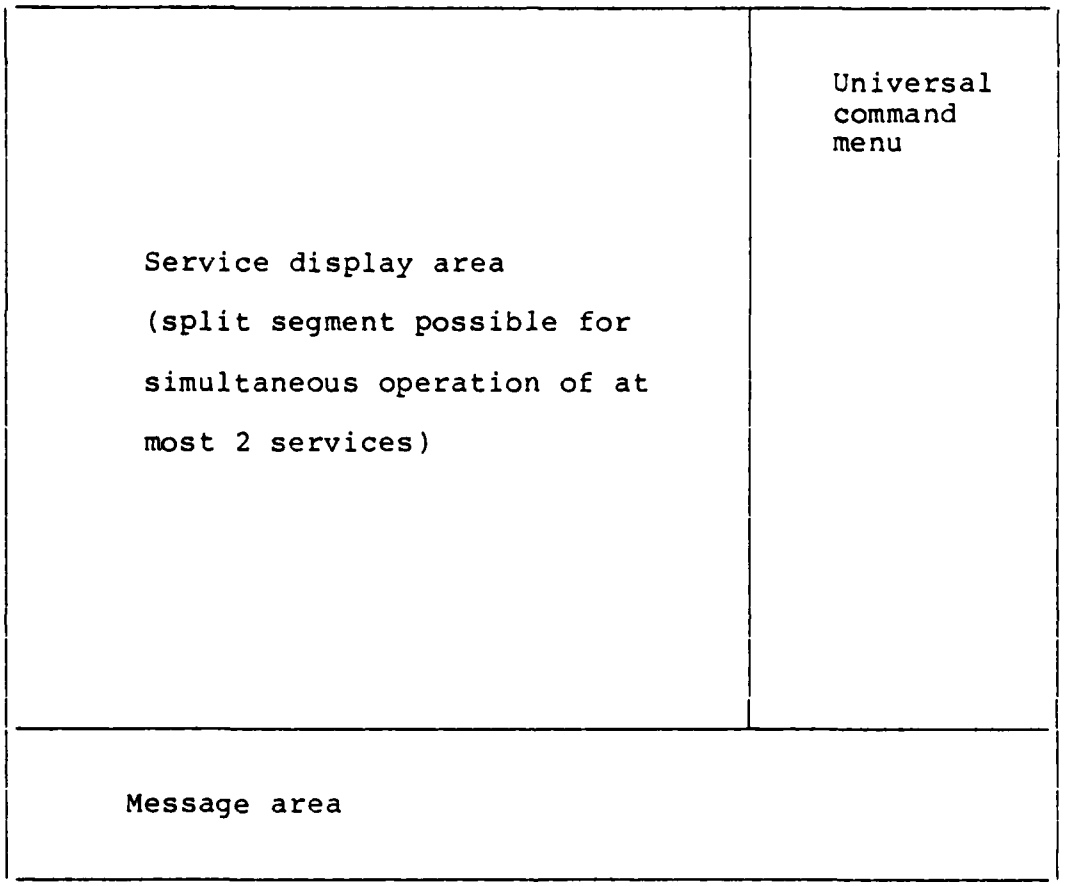


Figure 5: Fixed Segment Approach

estimation service).

Although the dynamic windowing approach offers more flexibility it also introduces more complexity into the software implementation. Therefore, the SAM/WS initially will be implemented using the fixed segment approach. However, the software will be designed such that a dynamic windowing capability can be retrofitted with minimum software implementation. If the microcomputer system chosen for the workstation offers windowing as an inherent feature, the dynamic windowing approach will be adopted from the start.

2.2 Scope of Domain Knowledge

The domain knowledge associated with software acquisition is very large. One of the key feasibility issues is whether a strategy can be devised to manage this domain knowledge such that a competent knowledge base can be designed for acquisition management. An approach that is very promising is to partition the domain knowledge into knowledge areas of a manageable size which can be treated relatively independently when designing the corresponding knowledge bases. The result of our Phase I analysis demonstrates that the software acquisition domain knowledge can be partitioned and implemented in approximately seventeen knowledge bases which fall into four classes. When so partitioned, the knowledge bases then

can be implemented incrementally. This partitioning will provide the capability to develop useful releases and a system growing in sophistication as more knowledge bases are added.

2.2.1 Domain Knowledge Partitioning

The software acquisition domain knowledge can be partitioned and implemented in seventeen separate knowledge bases. These knowledge bases are organized into four classes and are described below. The four classes of knowledge base defined are Tutorial, Checklist Prompting, Product Generating, and Parameter Generating.

a. Tutorial Knowledge Bases: This class of knowledge base provides tutorial information to the user. The tutorial information presented will depend on the user interactive context rather than the traditional approach of user directed search of information arranged in a hierarchy. There are two SAM/WS knowledge bases in this class:

(1) Software Acquisition Tutorial - provides tutorial on the Navy software acquisition process, the standard software engineering process, and software engineering environments.

(2) SAM/WS HELP - provides tutorial on the capabilities and operation of the SAM/WS.

b. Checklist Prompting: This class of knowledge base

provides prompting to the SAM. This prompting might be automatic based on the occurrence of pre-identified events or might occur as the result of user request on what to do next. There are four SAM/WS knowledge bases in this class:

(1) Software Acquisition Checklist and Management Plan - supports the user in developing a plan and checklist to guide him on a particular acquisition.

(2) Evaluation Criteria Checklist - supports development and management of a proposed evaluation checklist. Because evaluation will be based on the RFP requirements, this knowledge base will be related to the RFP knowledge base.

(3) Development Monitor and Control - supports the user during the software development phase (e.g., post contract award) by prompting for actions to be taken based on the development plan, budget, actuals, and milestones achieved.

(4) Test and Acceptance Monitor and Control - prompts the user on Quality Assurance (QA) checkpoints prior to Test and Acceptance (T&A) and on review actions during T&A.

c. Product Generating: This class of knowledge base will provide tailored outlines for necessary software acquisition documents and insert canned text where appropriate. The document then can be completed by the user through text entry and editing features of the SAM/WS.

There are nine knowledge bases in this class:

(1) Standards - provides a listing of requirements governing the software development process in the areas of guidelines, documentation, and support systems.

(2) Mission Element Need Statement (MENS) - supports the user in creating a MENS.

(3) Navy Decision Coordinating Paper (NDCP) - supports the user in creating an NDCP.

(4) Master Information Paper Summary (MINIMIP) - supports the user in creating a MINIMIP.

(5) Acquisition Strategy - supports the user in creating an acquisition strategy.

(6) Program Objectives Memorandum (POM) - supports the user in creating a POM.

(7) Statement of Work (SOW) - supports the user in creating a SOW.

(8) Deliverables - supports the user in creating a deliverables but in the form of a DD 1423.

(9) Request for Proposal (RFP) - supports the user in creating an RFP.

d. Parameter Generating: This class of knowledge base is used to support a user in the generation of complex input parameter sets used by SAM/WS analyses tools. It consists of two knowledge bases.

(1) Resource Estimation - aids the user in defining the necessary input parameter set to the resource estimation

tool.

(2) Schedule - aids the user in defining the input parameter set to the schedule analysis tool.

2.2.2 Knowledge Base Independence

Each knowledge base can be viewed as independent from the others. It can be designed, implemented, and used independent of considerations of the other knowledge bases. Therefore, the SAM/WS can be developed incrementally. A basic capability consisting of one or a few knowledge bases can be implemented in a short time. The SAM/WS then can be evolved over time as more and more knowledge bases are added. Although the knowledge bases can be developed separately and can be used as though they were independent, efficiency in operation can be improved by recognizing certain relations among the knowledge bases. Certain attributes are common to two or more knowledge bases. These attributes are dubbed "universal attributes." Examples of universal attributes are characteristics of the software system to be procured (e.g., size of system, whether or not it is real time in nature, and whether or not it supports a human interactive interface). These characteristics might be input or derived attributes of the Navy standards, resource estimation, or SOW knowledge bases. Therefore, if defined by the execution of one of these knowledge bases, it would

be desirable not to require the user to redefine them when using the other knowledge bases but rather to initialize these attribute values from the data base.

2.3 Software Design

Several issues relating to the design of the SAM/WS software were summarized in the Introduction. These issues are:

a. Functional Scope: Can expert system and traditional data processing analysis approaches be integrated in a single system?

b. Data Base: Can a data base support the specialized constructs of a knowledge base as well as the information analogous to that provided by a management information system?

c. Language: What is suitable for implementing the SAM/WS given that LISP has traditionally been used for expert systems, but that a language such as PASCAL might be better suited to the general software problem posed by SAM/WS?

Based on the analysis performed in the feasibility study, all of these issues have been satisfactorily resolved. The results of these analyses are discussed in the following subsections.

2.3.1 Software Architecture

Figure 6 presents an architecture for the SAM/WS software. The architecture is presented in the form of a uses hierarchy of information hiding modules arranged in levels of abstract machines. An information hiding module is a design component which defines a set of software functions while hiding the information about how the functions are implemented. This decomposition technique fosters the development of software which is designed for easily implemented enhancements and modifications. Each information hiding module is typically implemented by many programs. The uses hierarchy signifies that a program of an information hiding module can use only programs (functions) of information hiding modules beneath it in the hierarchy to perform its designated function. By utilizing this design technique, the SAM/WS software will be more easily evolved and extended than the more common approaches to design used today.

The information hiding modules are organized in five levels which can be viewed as abstract machines. Each abstract machine (hardware/software combination) provides functional capabilities to higher level software components that are more powerful and application-specific than the lower abstract machines levels. A brief description of each abstract machine level and the information

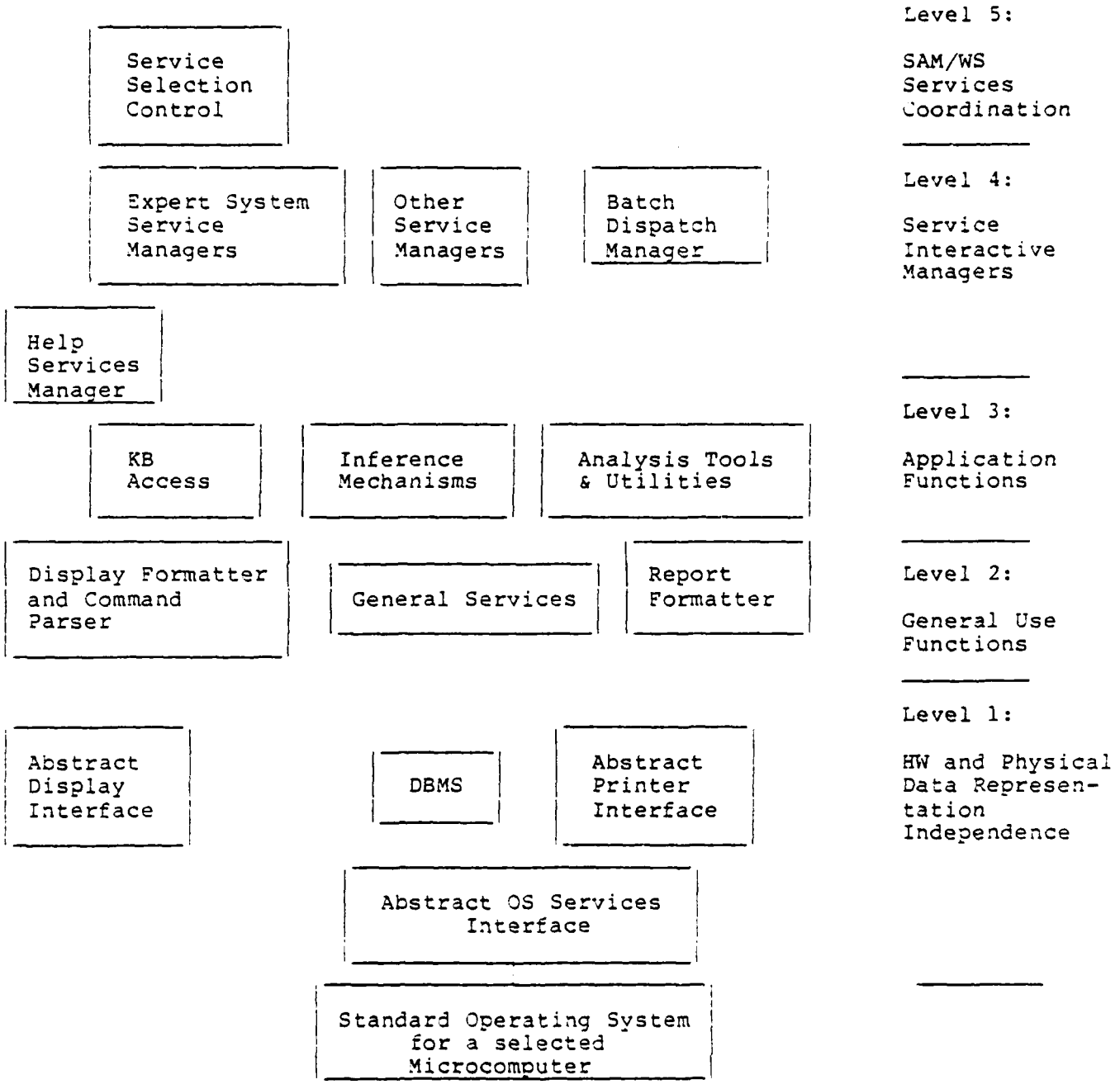


Figure 6: SAM/WS Software Architecture

hiding modules in each level is given below.

a. Level 1 - Hardware and Physical Data Representation Independence: This abstract machine provides functions which allow using programs (in higher level information hiding modules) to be implemented independent of any consideration of hardware characteristics or physical representation of data. It comprises the following information hiding modules:

(1) Abstract Display Interface: This module provides a display interface to using programs that are invariant under physical device changes.

(2) Abstract Printer Interface: This information hiding module provides a printer interface that is invariant to changes to the actual printer device.

(3) Data Base Management System (DBMS): This information hiding module provides using programs with a logical view of the data base concealing the physical storage format and location. It coordinates retrieval from remote data base appendages, as well.

(4) Abstract Operating System (OS) Services Interface: This information hiding module provides for OS independence through an invariant abstract OS services interface.

b. Level 2 - General Use Functions: This abstract machine provides functions which are of general use to higher level programs. It includes the following informa-

tion hiding modules:

(1) Display Formatter and Command Parser: This information hiding module formats CRT displays based on templates stored in the data base and directives from using programs. It parses commands returned from the Abstract Display.

(2) Report Formatter: This information hiding module controls the formatting of reports to be output on a line printer. It uses templates stored in the data base to generate the correctly formatted report.

(3) General Services: This information hiding module provides functions which are common to several higher level programs.

c. Level 3 - Application Functions: This abstract machine provides application-specific function which characterizes the SAM/WS. It comprises the following information hiding modules:

(1) KB Access: This information hiding module initializes knowledge bases from a variety of sources. It stores attribute values from interrupted or completed expert system interactive sessions.

(2) Inference Mechanisms: These information hiding modules provide inference mechanisms used by the various classes of Expert System Service Managers.

(3) Analysis and Utilities: These information hiding modules provide basic analysis tools and utilities

that might be accessed by Other Services Managers in an interactive mode, or that might be used as a batch job.

d. Level 4 - Service Interactive Managers: This abstract machine provides functions for the management of user interaction with the various SAM/WS services. It comprises the following information hiding modules:

(1) Expert System Services Managers: These information hiding modules control the various classes of expert systems (e.g., SOW creation or Software Acquisition Management Checklist). They control the expert system interaction with the user.

(2) Other Service Managers: These information hiding modules control the non-expert system interactive services such as resource estimation. They control the interaction between the user and the service function.

(3) Batch Dispatch Manager: This information hiding module coordinates batch processing initiated by the user, notifying the user of completion of batch jobs running in background mode.

(4) HELP Service Manager: This information hiding module provides a tutorial to the user on the capabilities and operation of the SAM/WS.

e. Level 5 - SAM/WS Services Coordination: This abstract machine provides overall management and coordination of the SAM/WS resources. It consists of a single information hiding module:

(1) Service Selection Control: This information hiding module controls the sign-on and session initiation process, coordinates service selection and interaction among services, and controls session termination.

There are several auxiliary functions not shown in Figure 6. These include tools to support knowledge base design, parsing, loading, and compiling. These functions also will be a part of the SAM/WS either integrated with the architecture shown in Figure 6 or implemented as stand-alone software applications.

The architecture presented in Figure 6 provides a comprehensive structure for the coexistence of expert systems and traditional analysis tools. Each is treated as a separate service. In those cases where an expert system requires a value generated by an analysis tool, this communication can be established through Service Selection Control acting as an intermediary in a manner transparent to the user.

The use of several information hiding modules as abstract interfaces will provide for device and OS independence. Coupled with the use of a high order language like PASCAL, this will maximize the portability of the SAM/WS software. In addition, the analysis has shown that the use of information hiding as the decomposition criteria fully supports and is compatible with an incremental development of SAM/WS software.

Efficient implementation of the SAM/WS software will be enhanced by the use of Knowledge Engineering, Inc. expert system products (e.g., KES) in the areas of Inference Mechanisms and Expert System Services Managers. These products are knowledge domain independent increasing the flexibility of the software for further evolution.

2.3.2 Data Base Considerations

The SAM/WS data base will contain all stored information generated by the SAM/WS and all knowledge bases associated with its expert systems. The knowledge base data structures will be relatively static in nature in that once entered they will be retrieved often but changed only when the acquisition expertise contained therein requires modification due to improved procedures or sophistication in methodologies. The other data (e.g., results of analysis runs, stored attribute values, and project status information) will be more dynamic, exhibiting frequent changes. The data base contents can be classified as:

a. Software Acquisition Products: The document product, such as an RFP or POM that result from a product generating expert system and which can be completed through text editing may be stored in any degree of completion.

b. Knowledge Bases: All the knowledge bases repre-

senting domain knowledge described in Section 2.2 will be stored in the data base.

c. Knowledge Base Execution Images: The attribute values of suspended or completed knowledge base interactions can be stored with descriptive labels. They may later be retrieved for continuation or review of the conclusions.

d. Universal Attributes: Certain knowledge base attributes are used by more than one knowledge base and/or by analysis tools. These attributes once initiated in value by one source should be made available for use by other programs to avoid the effort of deriving the attribute value anew.

e. Analysis Results: Results of analyses, such as resource estimation, can be labeled and stored for future reference and retrieval.

f. Project Status: Status on resource expenditure or milestone attainment can be stored and retrieved.

The various categories of data described above, with the exception of subparagraph b, can be partitioned according to project. The SAM/WS will allow access to the data of a given project only if the user is so authorized through a password protection system.

The SAM/WS data base is unique in that it integrates knowledge base data and dynamic acquisition management data. This uniqueness is most vividly exemplified by

the universal attributes within the data base. This class of data is a common point where the two types of dynamic data overlap.

Another unique aspect of the SAM/WS data base is that it can include as appendages parts of remote data bases (e.g., project status information residing in a Software Engineering Environment (SEE) data base hosted on a computer which will support communication with the SAM/WS). Such an appendage is an optional feature that may require SAM/WS-specific software components integrated in the SEE.

2.3.3 Implementation Language Strategy

Choosing an implementation language for the SAM/WS software is a complex problem. Several existing products will be used to implement the software (e.g., DBMS, expert systems, and resource estimation tools). These are written in a vartiety of languages including LISP and FORTRAN. Further, DOD is moving towards language standardization in Ada. Therefore, the problem to be addressed is how to make use of existing software components while moving in a direction that is compatible with DOD standards and that will maximize the portability of the SAM/WS software?

Two in-depth analyses were conducted to help resolve this problem. One surveyed the various dialects of LISP to evaluate the degree of difference in dialects and to

identify the best candidate if LISP is used as an implementation language. The other investigated the problems that might be expected if one attempted to convert to PASCAL existing expert system software written in LISP. The results of these analyses are summarized in Appendix D. Briefly, the differences in LISP dialects are enough to limit portability of programs written in LISP. Implementing expert systems in PASCAL is feasible but awkward and not cost effective when compared to LISP. The conversion costs for existing LISP programs to PASCAL would be substantial.

Based on the LISP to PASCAL conversion analysis and discussions with software professionals who have examined this question in regard to LISP to Ada conversion, it is felt that conversion to Ada, in the long term, is feasible for expert system software written in LISP.

If a mixed language strategy is adopted to accommodate existing software packages, the question becomes what is the most appropriate near-term language for new software implementation. Candidates are LISP, PASCAL, and FORTRAN. The overriding consideration in this regard is to design the SAM/WS software such that programs written in LISP can invoke FORTRAN or PASCAL programs. With this design, it would be desirable to make LISP the dominant language for any new software development in a mixed language system.

To make use of the limited time and budget available for DESAT Phase II, it is recommended that maximum use be made of existing software through a mixed language strategy with LISP as the dominant language supported by mechanisms to invoke programs written in PASCAL or FORTRAN which are portable. This approach will limit the initial portability of the software, but portability can be achieved in the long term by converting to Ada, when Ada becomes available, that portion of the system which is not portable.

2.4 Workstation Characteristics

The primary issue regarding the workstation hardware is whether a suitable microcomputer is commercially available. A microcomputer is deemed suitable if it is capable of supporting the SAM/WS software, provides the necessary packaged software, has available the necessary support software for development, and is priced appropriately.

The approach taken was to briefly summarize the requirements as outlined above. With these requirements as criteria, a survey and evaluation of the commerically available microcomputers was performed. This resulted in the identification of several microcomputers that are suitable to serve as the SAM/WS hardware.

2.4.1 Workstation Requirements

To support the SAM/WS software adequately, the hardware must provide certain minimum main memory, disk, and CPU characteristics. Based upon knowledge of modern micro-computer technology it was determined that speed would not be a limiting factor for a single station configuration, however insufficient main memory or disk storage could seriously impact both the development effort and performance of the system. Initial sizing analyses were made based on the software and knowledge base estimates which are presented in Section 2.5. The results of the disk storage sizing analyses are shown in Figure 7.

Software Source Code	3.2 MB
Software Object Code	2.9 MB
Knowledge Base Source Code	1.0 MB
Knowledge Base Object Code	1.5 MB
Other Data Base	.5 MB
TOTAL	9.1 MB

Figure 7: Disk Storage Sizing

Based on these estimates and a knowledge of memory requirements, the recommended system memory capacity is:

Main Memory - 1 megabyte (MB)

Disk Storage - 10 megabytes (MB)

This system memory capacity coupled with required SAM/WS response times justify a 16 bit word length microcomputer that supports the following hardware features:

a. Communications Port: A communications port which can be connected to a host computer is required.

b. Printer: A high quality printer is required for the production of reports and documents.

c. CRT: A high quality desk top CRT and keyboard. (A bit mapped display for graphics and a cursor or other pointing device is desirable.)

d. Disk Drive: A disk drive with greater than 10 megabytes of storage to support a full capability SAM/WS. A floppy disk with greater than 1 megabyte of storage is acceptable for early development of the SAM/WS.

In addition, the microcomputer must have available an OS and DBMS as well as LISP, PASCAL, and FORTRAN compilers/ interpreters, text editor, linkers, and library support for software development and life cycle support.

In order for the operational replication of the SAM/WS in the Navy to be feasible, the unit cost for the hardware should be approximately $10,000. Given the trend in hardware cost performance we estimated that a system costing

less than $50,000 today will be in that range by the end of the SAM/WS development period.

In summary, the minimal requirements used to evaluate workstation hardware were:

1. Supports hard disk (10 megabytes), printer, CRT, and communications.
2. 16 bit word length with a minimum of one megabyte main memory.
3. Costs under $50,000.
4. Has available an OS, DBMS, LISP, PASCAL, FORTRAN, text editor, linkers and library support.

2.4.2 Survey

The list of microcomputers surveyed is shown in Figure 8. Of these, only four are suitable candidates for the SAM/WS:

Vendor	Model
Alpha Micro	AM-1030
Apollo Computer	Domain
Three Rivers Computers	Perq
Wicat	System 150

Of these candidates, only the AM-1030 meets all the requirements today. The other three will meet the requirements upon release of pending hardware or software features.

VENDOR	MODEL	ISA	COST ($000)	LISP	MAX MEMORY	ACCEPT
Action Computer Enterprize	DPC-186 with Discovery	8086	17	none	1 Mbyte	no
Alpha Micro	AM-1030	N/A	15-30	Alpha Lisp	2 Mbytes	yes
Altos	ACS 8000	Z80	8-13	TLC Lisp	256 Kbytes	no
	ACS 8600	8086	10-15	none	1 Mbyte	no
Apollo Computer	Domain	68000	24-40	PSL and T-Lisp	1 Mbyte	yes*
Apple	Apple II	6502	<5	none	64 Kbytes	no
	Apple III	6502	<7	none	128 Kbytes	no
Cromemco	CS-3	Z80	17	Cromemco Lisp	64 Kbytes	no
DEC	PDP 11/23 Plus	LSI-11	22-30	**	1 Mbyte	no
Fortune	Fortune	N/A	15	none	1 Mbyte	no
Hewlett Packard	HP-1000	HP2100	7-15	none	2 Mbytes	no
Intel	System 86/330	8086	19	none	1 Mbyte	no
Lisp Machine Inc	CADR Model A	N/A	70	LM Lisp	4 Mbytes	no
	Lambda	N/A	45-60	LM Lisp	4 Mbytes	no
Micro DaSys	Micro DaSys	68000	12	N/A	256 Kbytes	no

*Pending the release of H/W or S/W under development
**PDP 11 series Lisps can only address 64Kbytes of memory
N/A - Information not available

Figure 8: Summary of Surveyed Workstation Hardware and Software

VENDOR	MODEL	ISA	COST ($000)	LISP	MAX MEMORY	ACCEPT
Radio Shack	Model 16	68000/Z80	5-10	none	512 Kbytes	no
Renaissance	RSI System	68000	6-13	none	4 Mbytes	no
Symbolics	LM-2	N/A	99	LM Lisp	4 Mbytes	no
	3600	N/A	75	LM Lisp	8 Mbytes	no
Three Rivers Computers	Perq	N/A	36	Spice Lisp	1 Mbyte	yes*
Western Digital	nu Machine	N/A	25	N/A	32 Mbyte	no
Wicat	System 100	68000	8-22	PSL	6 Mbyte	yes*
	System 150	68000	8-21	PSL	1.5 Mbyte	yes*
Xerox	1100 (Dolphin)	N/A	60	Interlisp-D	1 Mbyte	no
Zilog	Z8000	Z8000	15	none	2 Mbyte	no

*Pending the release of H/W or S/W under development
**PDP 11 series Lisps can only address 64Kbytes of memory
N/A - Information not available

Figure 8: Summary of Surveyed Workstation Hardware and Software (continued)

The Perq of Three Rivers Computers offers some advantages beyond the stated requirements. It is being purchased by the Navy for installation aboard the USS Carl Vinson (CVN-70). Carnegie-Mellon University is under contract to develop software for the Perqs installed on the USS Carl Vinson. Much of this software may have application in the SAM/WS. Conversely, the domain-independent expert systems developed for the SAM/WS may have application aboard ship. Therefore, the Perq may have some added advantages over the other three candidates.

2.5 Development Strategy

To implement a fully capable SAM/WS is a large undertaking both in terms of the development effort involved and the research required. In order to maximize the early availability of operationally useful components, an incremental approach is appropriate. Such an incremental approach will result in several successive operational releases leading to a fully capable SAM/WS. Our experience indicates that as many as four or five such releases might be required. For the purposes of this feasibility effort, only the first release and the full system are addressed. It is a reasonable goal of DESAT Phase II to implement the first release.

2.5.1 Release 1 Description

The first implementation increment of the SAM/WS, Release 1 is sized to fit the expected resources provided by the DESAT Phase II program. A summary of the functional capabilities of Release 1 contrasted to the functional capabilities of the full SAM/WS, as described in Section 1.2, is presented in Figure 9. Release 1 will use all of the full system hardware except the communications interface to a remote computer. The Release 1 SAM/WS software will be implemented to support Requirements and Acquisition Preparation activities and to fully demonstrate the feasibility of the software architecture, user interface, and partitioning of the knowledge base in the expert system. As shown in Figure 9, the knowledge bases for the standards, SOW, and Deliverables will be fully implemented. The Acquisition Tutorial and SAM/WS HELP will be partially implemented.

Hardware:	Release 1	Full System
16 bit p (1 Meg mem)	x	x
Printer	x	x
CRT Terminal	x	x
Communications		x
Disk (10 Meg)	x	x
Supported Acquisitions Activities:		
POM Preparation		x
Requirements	x	x
Acquisition Package	x	x
Non-competitive Acquisition Package		x
In-house Acquisition Package		x
Proposal Evaluation		x
Development Oversight		x
Test & Acceptance		x
Production & Delivery		x
In-service Engineering		x
Software Capabilities		
Interactive Interface	x	x
Expert system knowledge bases		
Acq. Tutorial	P	x
SAM/WS Help	P	x
Acq. Checklist		x
Evaluation Checklist		x
Dev. Monitor & Control		x
T&A Monitor & Control		x
Standards	x	x
MENS		x
NDCP		x
MINIMIP		x
Acquisition Strategy		x
POM		x
SOW	x	x

P = partial implementation

x = full implementation

Figure 9: Release 1 versus Full System Capabilities

Software Capabilities (con't.)	Release 1	Full System
Deliverables	x	x
RFP		x
Res. Estimation		x
Schedule		x
Analysis Tools		x
Text Editor	x	x
Report Generator	P	x
Data Base Management	x	x
Operating System	x	x
Utilities	P	x

P = partial implementation

x = full implementation

Figure 9: Release 1 versus Full System Capabilities (continued)

2.5.2 Resource Estimates

The development manpower resource estimates for Release 1 and the Full System are based upon the prior experience of our team in developing the proprietary expert system product, Knowledge Engineering System (KES), coupled with the knowledge gained as a result of our feasibility analyses and implementation of the Standards Demonstration Expert System during DESAT Phase I. These manpower resource estimates are considered low risk and provide a detailed breakdown of the effort required to implement the Development Strategy described in Section 2.5. The manpower resource estimates are divided into two types with the first type consisting of expert system professionals who are primarily responsible for software development in terms of performance specification, design, and production of source lines of code (SLOC). The second type consists of domain knowledge (e.g., Navy Standards) professionals who are primarily responsible for knowledge base development in terms of producing lines of text (LOT). The manpower resource breakdown presented below describes the effort in man-months to develop Release 1 and the additional effort in man-months to provide the Full system based upon using the completed version of Release 1. Therefore, manpower resources in man-months (MM) for the development of the Full System is obtained by the formula:

FS = R1 + FSAE

where

FS is Full System MM

R1 is Release 1 MM

FSAE is Full System Additional Effort MM

with MM = MM for SLOC + MM for LOT

a. Software Development: The software development manpower in man-months (MM) for source lines of code (SLOC) was estimated using the following formula:

MM for SLOC = K x SLOC PF

where

"K" is a constant multiplier of 1.2 representing management and administrative overhead.

"SLOC" is Source Lines of Code excluding comments.

"PF" is Productivity Factor in terms of expected production rate for lines of source code per man-month.

The constant K is taken as 1.2, which has proven to be appropriate for a 5-10 person project. The productivity factor will vary with the difficulty of the code involved. Based on the results of the feasibility analysis, three levels of production difficulty are evident for SLOC as follows:

Difficulty	Productivity Factor (PF) (SLOC per MM)
easy	400
medium	300
hard	200

Figure 10 summarizes the size of each software component in source lines of code (SLOC) for Release 1 (R1) and the Full System Additional Effort (FSAE) and shows the results of the computation

(SLOC PF) rounded to the nearest tenth.

Taking the totals for this computation in Figure 10 and multiplying them by K (1.2) yields the following for software development manpower:

Release 1 (R1) is 39.4 MM for SLOC.

Full System Additional Effort (FSAE) is 127.0 MM for SLOC.

The results above and in Figure 10 for the FSAE represent the additional resources assuming Release 1 is complete. Therefore, the total software development resources for the Full System is 166.4 MM for SLOC.

b. Knowledge Base Development: The knowledge base development manpower resources in MM for LOT were estimated based upon our experience in developing the Navy Standards Demonstration Expert System and consultation with professionals who were experts in the knowledge bases identified

Software Component	PF	Release 1			FSAE		
		SLOC	SLOC	PF	SLOC	SLOC	PF
Modules (see Section 2.3):							
Service Selection Control	400	200		.5	500		1.3
Expert Systems Service Mgrs.	-	Note 1		-	Note 1		-
Other Service Mgrs.	400	500		1.3	2000		5.0
Batch Dispatch Mgr.	400	-		-	500		1.3
Help Service	300	500		1.7	1000		3.3
Display Formatting	300	2000		6.7	4000		13.3
Abstract Display	200	2000		10.0	3000		15.0
KB Access	-	Note 1		-	Note 1		-
Inference Mechanisms	-	Note 1		-	Note 1		-
Analysis & Utilities	400	500		1.3	15000		37.5
Report Formatter	300	1500		5.0	3000		10.1
Abstract Printer	300	1500		5.0	1500		5.0
DBMS	300	Note 2		-	2000		6.7
Auxiliary Software	400	500		1.3	3000		7.5
Total				32.8			105.9

NOTE 1: These components off-the-shelf from Knowledge Engineering, Inc.

NOTE 2: An existing DBMS will be used unmodified for Release 1.

Figure 10: Software Development Resource Analysis for MM of SLOC

in Section 2.2. The MM estimates for LOT were derived based on the rationale that development of LOT has the same productivity factor (PF) for each of the knowledge bases in SAM/WS and the K constant of 1.2 is not appropriate due to the partitioning of the data bases. Using this rationale, an appropriate estimate of the MM for LOT of each knowledge base described for the SAM/WS in Section 2.2 was performed using the following information.

(1) There were 704 LOT developed for Navy standards demonstration knowledge base and it required six weeks of effort.

(2) The complete Navy standards knowledge base will be approximately 3000 LOT which shows that the 704 LOT were about 25% of the effort.

(3) Based upon subparagraphs (1) and (2) above, the development rate for LOT is 6 MM per 3000 LOT.

(4) The size of the Navy standards knowledge base provides the basis for estimating the size of other knowledge bases (e.g., the RFP knowledge base is twice as large, 6000 LOT, as the Navy standards knowledge base).

Based upon the above information, Figure 11 described the LOT for each knowledge base and the associated MM for LOT production that is required to develop Release 1 (R1) and the FSAE leading to a Full System (FS). As shown in Figure 11, R1 is 16.5 MM for LOT and FSAE is 79.2 MM for LOT; therefore, FS is 95.7 MM for LOT.

Taking the sum of the MM for SLOC and LOT and using the formula (FS = R1 + FSAE) in Section 2.5.2, the following is the breakdown of development manpower estimates for R1, FSAE and FS.

R1 = 39.4 MM for SLOC + 16.5 MM for LOT

R1 = 55.9 MM

FSAE = 127.0 MM for SLOC + 79.2 MM for LOT

FSAE = 206.2 MM

FS = 262.1 MM

Knowledge Base	R1		FSAE	
	LOT	MM*	LOT	MM*
Software Acquisition Tutorial	1500	3.0	9000	18.0
SAM/WS Help	900	1.8	1500	3.0
Soft.Acq.Checklist & Mgt. Plan	-		2100	4.2
Evaluation Criteria Checklist	-		3000	6.0
Development Monitor & Control	-		6000	12.0
T&A Monitor & Control	-		4500	9.0
MENS	-		1500	3.0
NDCP	-		1500	3.0
MINIMIP	-		1500	3.0
Acquisition Strategy	-		1500	3.0
POM	-		1500	3.0
SOW	3000	6.0	-	-
Deliverables	600	1.2	-	-
RFP			6000	12.0
Navy Standard (704 LOT Complete)	2250	4.5	-	-
Total (MM)		16.5		79.2

*6MM per 3000 LOT

Figure 11: Knowledge Base Resource Analysis of MM for LOT

3. CONCLUSIONS

Although the SAM/WS will require a substantial development effort, all aspects of such a workstation are feasible. The magnitude of the development effort can be made manageable through an incremental development strategy. Such a strategy is feasible because the software acquisition domain knowledge and the SAM/WS software are amenable to partitioning. This allows development of the SAM/WS in a series of operationally useful releases of increasing functional capabilities. Release 1 can be implemented within the confines of DESAT Phase II to provide the framework that will support a full capability SAM/WS as well as furnish to the Navy acquisition managers a useful tool to improve and support the accomplishment of their responsibilities.

The Navy standards expert system demonstration showed that a portion of the software acquisition knowledge domain can be implemented as an independent partition. It also provided the basis for estimating the size of all knowledge bases and the associated software development resources. The state-of-the-art literature survey further corroborated the practicality of implementing a software acquisition expert system with knowledge base domain-independent software. In the knowledge base area, further analysis will be required in Phase II to determine the best approach for

implementing universal attributes. The SAM/WS software architecture will provide all the necessary functions to support software acquisition management while exhibiting flexibility for the incorporation of evolving function. It provides for domain independence of the expert system software components making the architecture a suitable base for applications other than software acquisition management. The architecture also provides the framework for a well human-engineered user interface that is simple to use, varied in console modes but uniform in the syntax and semantics presented to the user. This will allow people with no computer experience to interact in a simple manner while more experienced users might use more sophisticated techniques to efficiently access the advanced features of the workstation.

Further design analysis is required to determine the best approach to the data base design allowing remote data base appendages. This design will depend on the architecture of the computer software to some extent. The implementation language analysis indicates that the most practical strategy for DESAT II is a mixed language design with LISP invoking FORTRAN or PASCAL programs. For the long term, conversion to Ada is feasible for conversion of previously non-portable programs to portable programs.

The sizing analysis for the SAM/WS indicates that it can be hosted on today's 16 bit microprocessors. Several

adequate commercially available candidates were identified. One candidate, Perq of Three Rivers Computers, is particularly attractive since it is being used for another Navy project on which software applicable to the SAM/WS is being developed.

Implementation of an operationally effective SAM/WS is feasible. The development effort required to totally implement SAM/WS is estimated to be twenty-two man-years over a three year period. An operationally significant subset which supports easy expansion to a full capability SAM/WS can be produced within one year under the DESAT Phase II effort.

4. RECOMMENDATIONS

The difficulties encountered today in developing and maintaining software are critical to the Navy because they can impact fleet readiness. Steps must be taken to overcome the current problems. Application of expert system technology to the support of Navy software acquisition managers will have a significant positive impact on software quality and development lead time. Because of the importance of this problem to the Navy, the following recommendations are submitted for SAM/WS development.

- Fund an initial operationally useful subset of the SAM/WS (Release 1) which supports most essential near-term capabilities such as software acquisition tutorial, HELP, Navy standards, SOW and deliverable preparation. (Initial SAM/WS (Release 1) effort is 55.9 man-months and is within the scope of the DESAT Phase II program.)
- Fund the 206.2 man-months for the Full System Additional Effort (FSAE) over a two or three year period after completion of the initial SAM/WS (Release 1) development. (Design of SAM/WS facilitates evolution to a full capability through incremental funding for the 206.2 man-months that are required for completion.)

APPENDIX A - State-of-the-art Survey

During the last decade a very large number of expert systems have been developed and evaluated. This appendix provides a brief introductory survey to this work with several examples of implemented systems. The material is divided into four sections: (1) unifying concepts; (2) systems developed in a research environment; (3) military and industrial applications; and (4) key conclusions including projections for the near future. For brevity, several related topics are considered beyond the scope of this survey (answer justification, metaknowledge, natural language interfaces, learning/adaptive systems, domain-independent systems, performance assessment, etc.) and therefore not addressed in this appendix.

A.1 Unifying Concepts

While a great variety of methods have been used to construct expert systems, such systems can typically be viewed as composed of two distinct parts:

a. Knowledge Base: Data structures representing the system's "knowledge" about how to solve a class of problems; and

b. Inference Mechanism: Programs that interpret the knowledge base in the context of a specific problem to generate useful information for the user.

For example, in the SAM/WS the knowledge base includes information about procurement policies, NAVY standards, software engineering methodologies, etc. The inference mechanism is the program that applies this abstracted knowledge to support decisions made by a software acquisition manager.

Several approaches have been taken to implementing these two components of expert systems, and these will be referred to repeatedly in subsequent sections. Statistical pattern classifers are expert systems where the knowledge base consists of tables of probabilities, and the inference mechanism is a program that calculates a new set of probabilities about relevant outcomes (e.g., using Bayes Theorem). Rule-based systems use a knowledge base of conditional rules ("if A is true then B is true") and have an inference mechanism called a rule interpreter (for example, deduction may be used). Network-based systems use a knowledge base of nodes (concepts or frames) which are interconnected by links (relationships). Inference mechanisms used in network-based systems have been viewed from several perspectives: as "activation" of nodes by "signals" on incoming links (Hayes-Roth, 1978), as a model of human hypothetico-deductive reasoning (Reggia, 1982), etc. Rule-based systems are the most likely choice for the SAM/WS although network-based systems may have limited application as well.

A.2 Survey of Research Systems

The range of problems to which expert systems have been applied during the last decade in a research setting is quite impressive. The following implemented systems, grouped by area of application, are intended to be representative but certainly not exhaustive examples of the state-of-the-art.

Medicine

Medical expert systems have been studied for over twenty years, with systems being developed to assist with diagnosis, treatment, disease staging, and predicting prognosis. See (Reggia, 1982) for a recent review. Several examples can be given of systems that perform at the level of medical experts or even better: diagnosis of acute abdominal pain using statistical pattern classification (deDombal, 1975); treatment of infectious diseases using rule-based deduction (Shortliffe, 1976) and the diagnosis and treatment of glaucoma using a combination of the rule-based and network-based approaches (Weiss, 1978). In addition, a number of network-based models of human diagnostic reasoning have been implemented (e.g., Pople, 1975; Pauker, 1976/ Reggia, 1981). These latter systems are still experimental, but they have clearly demonstrated the potential to handle problem-solving tasks that would be very difficult to attempt with other techniques.

Law

Work on expert systems in law has not received as much attention as in medicine, and at present it is generally a less explored field. However, several relevant examples still can be cited. A system called LRS has been developed to aid lawyers and legal assistants with the retrieval of documents based on their relevance to specific legal problems (Hafner, 1978). This is analogous to the problem of standards selection, a task to be included in the SAM/WS which we implemented during Phase I of this project. Another legal expert system has been implemented to assist with legal document preparation (Sprowl, 1979). This system is relevant to the proposed SAM/WS as it combines expert system techniques with automatic report generation. In field tests, this legal documentation system was quite successful. Other legal expert systems include: rule-based analysis of product liability litigation (Waterman, 1980), simulation of proposed legislation to assess its impact (Cook, 1978), network-based analysis of the tax consequences of corporate transactions (McCarty, 1980), and analysis of civil battery and assault cases (Meldman, 1977).

Chemistry and Biology

A number of expert systems have been developed in these fields. For example, one of the earliest rule-based systems analyzed mass spectra to elucidate the

structure of organic molecules (Buchanan, 1969). This system is currently in use by chemists around the world (Economist, 1982). More recently a similar system has been implemented for inferring DNA structures from restriction enzyme segmentation data (Stefik, 1978). Still another rule-based system has been implemented for interpretation of protein x-ray crystallographic data (Nii, 1978).

Engineering

While at least one expert system has been developed to assist mechanical engineers (Bennett, 1978), most engineering applications of expert system technology has been directed towards electronic problems. Computer-aided design of electronic systems has been a popular area for this research, and has included work on both analog (Sussman, 1977) and digital (Grinberg, 1980) circuit design. Another area that has received recent attention is the localization of circuit failures in electrical devices (Brown, 1977).

Automatic Programming

Automatic programming (or program synthesis) is the task of combining the specifications of a desired goal with knowledge of algorithmic processes, design techniques, and computer languages to produce a machine executable program. A number of expert systems have been developed for automatic programming where the pro-

gram's "goal" is specified either in natural language or through input-output examples (Barstow, 1977; Biermann, 1979; Green, 1977; Manna, 1977). Some of these systems have achieved an impressive level of performance.

Others

A wide variety of additional expert system examples could be cited, ranging from geological evaluation of exploration sites for their mineral potential (Duda, 1977) to modelling skill acquisition by student pilots (Goldstein, 1977). The above discussion demonstrates that expert system technology has been successfully applied to a wide range of disciplines to reduce work-load and improve the performance of professionals.

A.3 Survey of Applications

During the last two or three years there has been a surge of interest by business, industry and the military in the potential applications of expert system technology. Expert systems developed in this context differ from those described above in that they are intended from the start to be used when they are implemented for commerical/ industrial/military problem solving in the field.

One popular problem to which expert system technology has been applied involves the diagnosis of the cause of failure in complex systems. IBM has implemented an expert system that advises its field personnel about the diagnosis

of faults occurring in computer installations (Bennett, 1981). This rule-based system identifies specific software and/or hardware components most likely to be responsible for an observed fault and offers a brief explanation of the factors supporting these indictments. Similar systems are under study at the Jet Propulsion Laboratory (Friedman, 1982) for automatic diagnosis of spacecraft malfunctions, and at Satellite Business Systems for ground/satellite failure analysis.

A wide variety of other commerical/industrial applications exist. Major oil companies such as Schlumberger are using KBESs to interpret geological data from oil-well measurement devices. Others, (e.g., Amoco) are applying expert systems to assist geologists with the complex task of evaluating the commercial potential of exploration sites (Economist, 1982). The Digital Equipment Corporation (DEC) has developed a large, rule-based system that configures complex VAX-11 computer systems for individual customers (McDermott, 1981). Westinghouse, Texas Instruments, Fairchild and Hewlett Packard are developing expert systems that capture manufacturing knowledge and aid in the design process (Economist, 1982).

Finally, a number of military applications of expert systems have recently become evident. For example, a statistical pattern classification system has been developed by the Navy for diagnosis of acute abdominal

pain. It is anticipated that the use of this system will significantly reduce unnecesaary evacuations of sailors aboard nuclear submarines where physicians are not available and radio silence must be maintained (Henderson, 1978). The Air Force is developing an experimental expert system to support tactical air mission planning (Engleman, 1979). An extensive review of potential expert system technology applications to tactical command and control has recently been published (Wohl, 1981).

A.4 Conclusions

Based on the above survey, as well as our own experience at Software A&E, we feel that a number of conclusions are justified at the present time.

a. Expert system technology has been applied to a wide variety of problems, with several empirical demonstrations of its effectiveness.

b. This technology has matured to the point where it is now being applied to industrial and military tasks in support of decision making. This recent emergence of applications is consistent with the observation that expert systems are becoming cost-effective.

c. It is essential that the domain expert/application specialist as well as intended users be involved with the design and implementation of an expert system.

d. A variety of methods have evolved for representing

and processing knowledge in expert systems. Contrary to what is sometimes stated in the literature, there is no single best method because each has certain advantages and disadvantages. Having a variety of methods to choose from or to combine is very useful during expert system construction.

e. The recent emergence of domain-independent tools (such as KES) for building expert systems is a major technological advance in making these systems cost-effective.

f. Expert systems must not only be able to generate "answers," but must also be able to justify those answers as well as explain the inference process used in reaching them. Along with other factors that lead to a "friendly" system, this is important for user acceptance and confidence.

g. In the near future a variety of factors (such as more advanced knowledge representation methods, the emergence of domain-independent systems, and falling hardware costs) will act together to improve the cost-effectiveness of expert system technology. Based upon these facts, we anticipate a very rapid growth in the application of expert systems during the next decade.

A.5 References

Barstow, D, "A Knowledge-Based System for Automatic Program Construction," IJCAI-5, 1977, pp. 382-388.

Bennett, J., Creary, L., Englemore, R., and Melosh, R., "SACON - A Knowledge-Based Consultant for Structural Analysis, Technical Report." Computer Science Dept., Stanford University, Sept. 1978.

Bennett, J. and Hollander, C., "DART - An Expert System for Computer Fault Diagnosis," IJCAI-7, 1981, pp. 843-845.

Bierman, A. and Smith, D., "A Production Rule Mechanism for Generating LISP Code," IEEE Trans. on Systems, Man and Cybernetics, 9, 1979, pp. 260-276.

Brown, A., "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures." AI-TR-362. MIT, March 1977.

Buchanan, B., Sutherland, G., and Feigenbaum, E., "Heuristic DENDRAL," Machine Intelligence, 4, 1969.

Cook, S. and Stamper, R., "LEGOL as a Tool for the Study of Bureaucracy," in The Information Systems Environment, ed. H. Lucas. North Holland, Amsterdam, 1978.

deDombal, F., "Computer Assisted Diagnosis of Abdominal Pain," Advances in Medical Computing, eds. J. Rose and J. Mitchell, Churchill-Livingston. New York, 1975, pp. 10-19.

Duda, R.; Hart, P., Nilssen, N.; Reboh, R., et. al., Development of a Computer-Based Consultant for Mineral Exploration. Annual Report. Stanford Research Institute, Menlo Park, California, October 1977.

"Economist: Why Can't a Computer Be More Like A Man," The Economist, Jan. 9, 1982, pp. 77-81.

Engleman, C., Berg, C. and Bischoff, M., "KNOBS - An experimental Knowledge-Based Tactical Air Mission Planning System and a Rule-Based Aircraft Identification Simulation Facility," IJCAI-6, 1979, pp. 247-249.

Friedman, L., "Jet Propulsion Laboratory Automated Problem Solving Group," SIGART Newsletter, 79, 4, Jan. 1982.

Goldstein, I. and Grimson, E., "Annotated Production Systems A Model of Skill Acquisition," IJCAI-5, 1977, pp. 311-316.

Green, C., "A Summary of the PSI Program Synthesis System," IJCAI-5, 1977, pp. 380-381.

Grinberg, M., "A Knowledge-Based Design Environment for Digital Electronics." Ph.D. Dissertation. Dept. of Computer Science, University of Maryland, College Park, Maryland, Sept. 1980.

Hayes-Roth, F., Waterman, D. and Lenat, D., "Principles of Pattern-Directed Inference Systems," in Pattern-Directed Inference Systems, eds. F. Hayes-Roth and D. Waterman, Academic Press, 1978, pp. 577-601.

Hafner, C. D., "An Information Retrieval System Based on a Computer Model of Legal Knowledge." Ph.D. Thesis. The University of Michigan, Ann Arbor, 1978.

Henderson, J., Moeller, G., Ryack, B., and Shumaik, G., "Adaptation of a Computer-Assisted Diagnosis Program for use by Hospital Corpsman Aboard Nuclear Submarines," Proc. Second Ann. Symp. on Comp. Application. in Med Care, Nov. 1978, pp. 576-593.

Manna, Z. and Waldinger, R., "The Automatic Systnesis of Recursive Program," Proc. Symp. on Artificial Intelligence and Programming Languages, August 1977, pp. 29-36.

McCarty, L. and Sridharan, N., "The Representation of an Evolving System of Legal Concepts," Proc. 3rd Conf. of The Canadian Society for Computational Studies of Intelligence, Victoria, B. C., Canada, 1980, pp. 304-311.

McDermott, J. and Steele, B., "Extending a Knowledge-Based System to Deal with Ad Hoc Constraints," IJCAI-7, 1981, pp. 821-828.

Meldman, J., "A Structural Model for Computer-Aided Legal Analysis," Rutgers Journal of Computers and Law, 6, 1977, pp. 27-71.

Nii, and Feigenbaum, "Rule-Based Understanding of Signals," in Pattern-Directed Inference Systems, eds. F. Hayes-Roth and D. Waterman, Academic Press, 1978, pp. 483-502.

Pauker, S., et. al., "Towards the Simulation of Clinical Cognition," American Journal of Medicine, 60, 1976, pp. 981-996.

Pople, H., Myers, J., and Miller, R., "DIALOG: A Model of Diagnostic Logic for Internal Medicine," IJCAI, 1975.

Reggia, J., "Knowledge-Based Decision Support Systems - Development Through KMS," TR-1121. Dept. of Computer Science, University of Maryland, Oct. 1981.

Reggia, J., "Computer-Assisted Medical Decision Making," in Applications of Computers in Medicine, ed. M. Schwartz, IEEE Press, 1982.

Shortliffe, E., Computer-Based Medical Consultations: MYCIN, Elsevier, 1976.

Sprowl, J., "Automating the Legal Reasoning Process - A Computer that Uses Regulations and Statutes to Draft Legal Documents," American Bar Foundation Research Journal, 1979, pp. 1-81.

Stefik, M., "Inferring DNA Structures from Segmentation Data," Artificial Intelligence, 11, 1978, pp. 85-114.

Sussman, G., "Electrical Design - A Problem for Artificial Intelligence Research," IJCAI-7, 1977, pp. 894-900.

Waterman, D. and Peterson, M., "Rule-Based Models of Legal Expertise," Proc. 1st Ann. Nat. Conf. on Artificial Intelligence, 1980, pp. 272-275.

Weiss, S., Kulikowski, C. and Safir, ?., "Glaucoma Consultation by Computer," Computers in Biol. Med., 8, 1978, pp. 25-40.

Wohl, J., "Force Management Decision Requirements for Air Force Tactical Command and Control," IEEE Trans. on Systems, Man and Cybernetics, 11, Sept. 1981, pp. 618-639.

APPENDIX B - Standards Demonstration Expert System

As part of the feasibility analysis of an expert system workstation to support a Navy software acquisition manager, a partial knowledge base for software standards was implemented. The Knowledge Engineering System (KES) offered by Knowledge Engineering, Inc. was used to implement the standards expert system. This appendix contains example interactive sessions with the resulting expert system and a listing of the source statements representing the standards knowledge base.

Two examples of interactive sessions are presented. The first is a straight run through the expert system sequence to determine applicable standards. The second repeats a run through the sequence but with added commands to the expert system to justify its results and to display explanatory text. The input information typed by the user of the expert system is underlined to differentiate it from the expert system responses.

B.1 Example 1

The first example interaction is that of a software acquisition manager whose system of concern has the following characteristics:

- platform-based C^3 system
- multiple computers in the system
- acquisition phase is advanced development
- no firmware involved
- Navy standard computers
- multi-level security
- no interfaces to other systems.

```
@add kms*kms.PS-C

WELCOME TO KES.PS (2 26 82)
ENTER KES.PS KNOWLEDGE BASE:

@add kb.onr

KNOWLEDGE BASE LOADED-NO ERROR DETECTED

* THIS IS AN UNCERTIFIED KNOWLEDGE BASE *
THIS IS A DEMONSTRATION KNOWLEDGE BASE ON THE SUBJECT OF
SOFTWARE STANDARDS FOR DIGITAL SYSTEMS.  IT IS INTENDED
FOR USE BY NAVY SOFTWARE ACQUISITION MANAGERS.

THE KNOWLEDGE BASE IS ORGANIZED SUCH THAT STANDARDS
REQUIREMENTS DEFINITION WILL PROCEED IN THREE STAGES IN
THE FOLLOWING MANNER:

     STAGE 1 - DETERMINE GUIDELINES AND TOP LEVEL REQUIRE-
               MENTS AND NAVY-SPECIFIC REQUIREMENTS WHICH
               MIGHT APPLY TO YOUR SYSTEM.

     STAGE 2 - DETERMINE MANDATORY REQUIREMENTS FOR YOUR
               SYSTEM IN THE AREA OF DOCUMENTATION, HARD-
               WARE, OR SUPPORT SOFTWARE.

     STATE 3 - DETERMINE GUIDANCE SPECIFIC TO THE UNIQUE
               ASPECTS OF YOUR SYSTEM.

OK - FIRST ANSWER THE FOLLOWING QUESTIONS TO DETERMINE
WHICH STANDARDS APPLY TO YOUR SYSTEM.

     TYPE OF SYSTEM:
          (1)ADP
          (2)TACTICAL
          (3)NOT SURE
     =?
     3.

     OPERATIONAL BASE:
          (1)AIR OR SUBMARINE OR SURFACE SHIP
          (2)SHORE
     =?
     1.
```

```
     CLASSIFICATION:
          (1)ANTI-AIR WARFARE
          (2)ANTI-SUBMARINE WARFARE
          (3)ANTI-SURFACE WARFARE
          (4)STRIKE WARFARE
          (5)AMPHIBIOUS WARFARE
          (6)MINE WARFARE
          (7)SPECIAL WARFARE
          (8)MOBILITY
          (9)COMMAND CONTROL AND COMMUNICATIONS
          (10)INTELLIGENCE
          (11)ELECTRONIC WARFARE
          (12)NCO 2-6 '9 '18 '20
          (13)OTHER
     =?
     9.

THE TYPE OF SYSTEM HAS BEEN DETERMINED TO BE:

     TACTICAL

THE FOLLOWING LISTS OF GUIDELINES, TOP LEVEL REQUIREMENTS,
AND NAVY SPECIFIC AND DETAILED REQUIREMENTS ARE APPLICABLE
TO YOUR SYSTEM:

*** GUIDELINES ***

     DODD-5000-1 MAJOR SYSTEMS ACQUISITION
     DODI-5000-2 MAJOR SYSTEM ACQUISITION PROCESS
     DODI-5000-29 MANAGEMENT OF COMPUTER RESOURCES IN MAJOR
                  DEFENSE SYSTEMS
     DODI-5000-31 LANGUAGE STANDARDIZATION

*** TOP LEVEL REQUIREMENTS ***

     DOD-STD-480A CONFIG CONTROL ENGINEERING CHANGES
                  DEVIATIONS AND WAIVERS
     MIL-STD-481A CONFIG CONTROL ENGINEERING CHANGES
                  DEVIATIONS AND WAIVERS
     MIL-STD-490 SPECIFICATION PRACTICES

*** NAVY SPECIFIC AND DETAILED REQUIREMENTS ***

     SECNAVINST-3560-1 TACTICAL DIGITAL SYSTEMS DOCUMENTATION
                       STANDARDS
     NAVMATINST-4130-2A CONFIG MNGMT OF SOFTWARE FOR
                        TACTICAL SYSTEMS
     NAVMATINST-5200-27A PROCS FOR TRANSFER OF NAVY
                         TACTICAL SYSTEM SW RESP
     TADSTAND-A STANDARD DEFINITIONS FOR ECR IN TACTICAL
                DIGITAL SYSTEMS
```

TADSTAND-B STANDARD EMBEDDED COMPUTERS AND PERIPHERALS
TADSTAND-C LANGUAGE STANDARDIZATION POLICY FOR TACTICAL SYSTEMS
TADSTAND-D RESERVE CAPACITY REQUIREMENTS FOR TACTICAL DIGITAL SYSTEMS
TADSTAND-2 STANDARD SPECIFICATION FOR TACTICAL PROGRAM DOCUMENTATION
TADSTAND-3 REQUIREMENTS FOR INTER-DIGITAL PROCESSOR INTERFACE DOC
TADSTAND-9 SW QUALITY TESTING CRITERIA STANDARD FOR TACTICAL SYSTEMS
MIL-STD-1679 MILITARY STANDARD FOR WEAPON SYSTEM SOFTWARE DEVELOPMENT
MIL-S-52779A SOFTWARE QUALITY ASSURANCE PROGRAM REQUIREMENT
MIL-STD-1521A TECHNICAL REVIEWS AND AUDITS

OK - LET'S LOOK AT THE MANDATORY REQUIREMENTS FOR YOUR SYSTEM

MANDATORY REQUIREMENTS AREAS OF CONCERN:
(1)APPLICATION SOFTWARE DOCUMENTATION
(2)HARDWARE
(3)SUPPORT SOFTWARE
=?
1.

THE FOLLOWING LIST OF DOCUMENTS ARE MANDATORY FOR TACTICAL DIGITAL APPLICATION SOFTWARE. EACH DOCUMENT TITLE IS GROUPED WITH OTHER DOCUMENTS THAT ARE ACCEPTED AT APPROXIMATELY THE SAME TIME. NOTE THAT BEFORE ANY DOCUMENT IN A GROUP CAN BE ACCEPTED ALL OF THE DOCUMENTS IN THE PRECEEDING GROUP MUST BE ACCEPTED.

1. TYPE B SPEC OR SOD OR FOD

2. SOFTWARE DEVELOPMENT PLAN
   SOFTWARE QUALITY ASSURANCE PLAN
   SOFTWARE CONFIGURATION MANAGEMENT PLAN

3. PROGRAM PERFORMANCE SPECIFICATION
   INTERFACE DESIGN SPECIFICATION
   COMPUTER PROGRAM TEST PLAN
   COMPUTER PROGRAM TEST SPECIFICATION

4. PROGRAM DESIGN SPECIFICATION
   COMPUTER PROGRAM TEST PROCEDURES

5. PROGRAM DESCRIPTION DOCUMENT
   DATA BASE DESIGN DOCUMENT
   OPERATORS MANUAL
   SYSTEM OPERATORS MANUAL
   COMPUTER PROGRAM TEST REPORT
   PROGRAM PACKAGE DOCUMENT

THE FOLLOWING DOCUMENTS MAY BE REQUIRED AT VARIOUS POINTS THROUGHOUT THE SOFTWARE LIFE CYCLE:

SOFTWARE CHANGE PROPOSAL
SOFTWARE ENHANCEMENT PROPOSAL
SOFTWARE TROUBLE REPORT

OK - ANSWER THE FOLLOWING QUESTIONS TO OBTAIN GUIDANCE SPECIFIC TO THE UNIQUE CHARACTERISTICS OF YOUR SYSTEM.

```
MULTIPLE COMPUTER PROGRAMS:
     (1)YES
     (2)NO
=?
1.

ACQUISITION PHASE:
     (1)EXPLORATORY DEVELOPMENT OR ADVANCED DEVELOPMENT
     (2)ENGINEERING DEVELOPMENT
     (3)IN-SERVICE
=?
1.

FIRMWARE:
     (1)YES
     (2)NO
=?
2.

PROCESSOR TYPE:
     (1)NAVY STANDARD
     (2)NON-STANDARD
=?
1.

SECURITY:
     (1)MULTI-LEVEL SECURITY
     (2)SECURE COMMUNICATIONS
     (3)NONE
=?
1.

INTERFACES TO OTHER SYSTEMS:
     (1)YES
     (2)NO
=?
2.

MULTIPLE COMPUTERS:
     (1)YES
     (2)NO
=?
1.
```

THE FOLLOWING LIST PROVIDES SPECIFIC GUIDANCE ON REQUIREMENTS RELATIVE TO THE UNIQUE CHARACTERISTICS OF YOUR SYSTEM:

EITHER TYPE B SPECIFICATION OF MIL-STD-490 OR SOD OF SECNAVINST 3560.1 IS RECOMMENDED. ONE PPS FOR EACH

COMPUTER PROGRAM IS ALSO RECOMMENDED.

THE FOLLOWING MAY BE RELAXED FOR ADVANCED OR EXPLORATORY DEVELOPMENT:

-MIL-STD-1521A	AUDITS
-MIL-STD-1679	STATUS REVIEWS (5.12.3)
	INSPECTIONS AND AUDITS (5.12.4)
	QUALITY ASSURANCE (5.10.2)
	TESTING AT ALL LEVELS (5.8)
-TADSTAND 9	QUALITY ALSSURANCE

FOR SECURITY ENHANCEMENTS THE FOLLOWING ADDITIONS TO THE NAVY SPECIFIC STANDARDS SHOULD BE CONSIDERED:

- JCP PUB22 WWMCCS	ADP SYSTEM SECURITY MANUAL
- OPNAVINST 5510.1	DEPT OF NAVY INFORMATION SECURITY REGULATION
- OPNAVINST 5510.45	U.S. NAVY PHYSICAL SECURITY MANUAL
- NAVELEXINST 5510.2	GUIDELINES FOR COMPROMISING EMANATIONS CONTROL.

IDS'S ARE RECOMMENDED FOR INTERCOMPUTER INTERFACES, BUT ARE NOT MANDATORY.

READY FOR COMMAND:

B.2 Example 2

The following interaction is that of a software acquisition manager whose system of concern has the following characteristics:

- system is a component of NCCS - ashore
- single, non-standard computer
- single development agent
- PPS source information is available
- acquisition phase is in-service engineering
- the software will be implemented in firmware
- it will have interfaces to other systems.

next case.

READY FOR NEW CASE
THIS IS A DEMONSTRATION KNOWLEDGE BASE ON THE SUBJECT OF SOFTWARE STANDARDS FOR DIGITAL SYSTEMS. IT IS INTENDED FOR USE BY NAVY SOFTWARE ACQUISITION MANAGERS.

THE KNOWLEDGE BASE IS ORGANIZED SUCH THAT STANDARDS REQUIREMENTS DEFINTION WILL PROCEED IN THREE STAGES IN THE FOLLOWING MANNER:

STAGE 1 - DETERMINE GUIDELINES AND TOP LEVEL REQUIREMENTS AND NAVY-SPECIFIC REQUIREMENTS WHICH MIGHT APPLY TO YOUR SYSTEM.

STAGE 2 - DETERMINE MANDATORY REQUIREMENTS FOR YOUR SYSTEM IN THE AREA OF DOCUMENTATION, HARDWARE, OR SUPPORT SOFTWARE.

STAGE 3 - DETERMINE GUIDANCE SPECIFIED TO THE UNIQUE ASPECTS OF YOUR SYSTEMS.

OK - FIRST ANSWER THE FOLLOWING QUESTIONS TO DETERMINE WHICH STANDARDS APPLY TO YOUR SYSTEM.

TYPE OF SYSTEM:

(1)ADP
(2)TACTICAL
(3)NOT SURE
=?
3.

OPERATIONAL BASE:
(1)AIR OR SUBMARINE OR SURFACE SHIP
(2)SHORE
=?
2.

PART OF NCCS:
(1)YES
(2)NO
=?
1.

THE TYPE OF SYSTEM HAS BEEN DETERMINED TO BE:

TACTICAL

THE FOLLOWING LISTS OF GUIDELINES, TOP LEVEL REQUIREMENTS,

AND NAVY SPECIFIC AND DETAILED REQUIREMENTS ARE APPLICABLE TO YOUR SYSTEM:

*** GUIDELINES ***

DODD-5000-1 MAJOR SYSTEMS ACQUISITION
DODI-5000-2 MAJOR SYSTEM ACQUISITION PROCESS
DODI-5000-29 MANAGEMENT OF COMPUTER RESOURCES IN MAJOR DEFENSE SYSTEMS
DODI-5000-31 LANGUAGE STANDARDIZATION

*** TOP LEVEL REQUIREMENTS ***

DOD-STD-480A CONFIG CONTROL ENGINEERING CHANGES DEVIATIONS AND WAIVERS
MIL-STD-481A CONFIG CONTROL ENGINEERING CHANGES DEVIATIONS AND WAIVERS
MIL-STD-490 SPECIFICATION PRACTICES

*** NAVY SPECIFIC AND DETAILED REQUIREMENTS ***

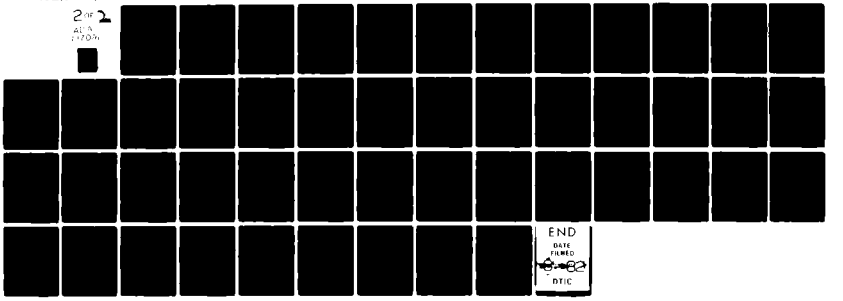
SECNAVINST-3560-1 TACTICAL DIGITAL SYSTEMS DOCUMENTATION STANDARDS
NAVMATINST-4130-2A CONFIG MNGMT OF SOFTWARE FOR TACTICAL SYSTEMS
NAVMATINST-5200-27A PROCS FOR TRANSFER OF NAVY TACTICAL SYSTEM SW RESP
TADSTAND-A STANDARD DEFINITIONS FOR ECR IN TACTICAL DIGITAL SYSTEMS
TADSTAND-B STANDARD EMBEDDED COMPUTERS AND PERIPHERALS
TADSTAND-C LANGUAGE STANDARDIZATION POLICY FOR TACTICAL SYSTEMS
TADSTAND-D RESERVE CAPACITY REQUIREMENTS FOR TACTICAL DIGITAL SYSTEMS
TADSTAND-2 STANDARD SPECIFICATION FOR TACTICAL PROGRAM DOCUMENTATION
TADSTAND-3 REQUIREMENTS FOR INTER-DIGITAL PROCESSOR INTERFACE DOC
TADSTAND-9 SW QUALITY TESTING CRITERIA STANDARD FOR TACTICAL SYSTEMS
MIL-STD-1679 MILITARY STANDARD FOR WEAPON SYSTEM SOFTWARE DEVELOPMENT
MIL-S-52779A SOFTWARE QUALITY ASSURANCE PROGRAM REQUIREMENT
MIL-STD-1521A TECHNICAL REVIEWS AND AUDITS

OK - LET's LOOK AT THE MANDATORY REQUIREMENTS FOR YOUR SYSTEM

MANDATORY REQUIREMENTS AREAS OF CONCERN:

```
          (1)APPLICATION SOFTWARE DOCUMENTATION
          (2)HARDWARE
          (3)SUPPORT SOFTWARE
     =?
     justify system type = tactical.

COMMAND ENCOUNTERED - DEFERRING CURRENT QUESTION
SUPPORTING EVIDENCE FOR THIS STATEMENT:

     TYPE4

EVIDENCE AGAINST THIS STATEMENT:

     *NONE*

READY FOR COMMAND:

display type4.

RULE NAME:  TYPE4

     IF TYPE OF SYSTEM =
          NOT SURE
       & OPERATIONAL BASE =
          SHORE
       & PART OF NCCS =
          YES
     THEN
       SYSTEM TYPE =
          TACTICAL

READY FOR COMMAND:

 justify guidelines = dodd-5000-1.

SUPPORTING EVIDENCE FOR THIS STATEMENT:

     GUIDELINE1

EVIDENCE AGAINST THIS STATEMENT:

     *NONE*

READY FOR COMMAND:

display guideline1.

RULE NAME:  GUIDELINE1

     IF SYSTEM TYPE =
          TACTICAL
```

AD-A117 091 SOFTWARE ARCHITECTURE AND ENGINEERING INC ARLINGTON VA F/G 9/2
SOFTWARE ACQUISITION MANAGER'S KNOWLEDGE-BASED EXPERT SYSTEM.(U)
JUN 82 A B FERRENTINO N00014-82-C-0130
UNCLASSIFIED NL



THEN
GUIDELINES =
DODD-5000-1 MAJOR SYSTEMS ACQUISITION
& DODI-5000-2 MAJOR SYSTEM ACQUISITION PROCESS
& DODI-5000-29 MANAGEMENT OF COMPUTER RESOURCES
IN MAJOR DEFENSE SYSTEMS
& DODI-5000-31 LANGUAGE STANDARDIZATION

READY FOR COMMAND:

continue.

CONTINUING PREVIOUS LINE OF QUESTIONING

MANDATORY REQUIREMENTS AREAS OF CONCERN:
(1)APPLICATION SOFTWARE DOCUMENTATION
(2)HARDWARE
(3)SUPPORT SOFTWARE
=?
1.

THE FOLLOWING LIST OF DOCUMENTS ARE MANDATORY FOR TACTICAL DIGITAL APPLICATION SOFTWARE. EACH DOCUMENT TITLE IS GROUPED WITH OTHER DOCUMENTS THAT ARE ACCEPTED AT APPROXIMATELY THE SAME TIME. NOTE THAT BEFORE ANY DOCUMENT IN A GROUP CAN BE ACCEPTED ALL OF THE DOCUMENTS IN THE PRECEEDING GROUP MUST BE ACCEPTED.

1. TYPE B SPEC OR SOD OR FOD

2. SOFTWARE DEVELOPMENT PLAN
SOFTWARE QUALITY ASSURANCE PLAN
SOFTWARE CONFIGURATION MANAGEMENT PLAN

3. PROGRAM PERFORMANCE SPECIFICATION
INTERFACE DESIGN SPECIFICATION
COMPUTER PROGRAM TEST PLAN
COMPUTER PROGRAM TEST SPECIFICATION

4. PROGRAM DESIGN SPECIFICATION
COMPUTER PROGRAM TEST PROCEDURES

5. PROGRAM DESCRIPTION DOCUMENT
DATA BASE DESIGN DOCUMENT
OPERATORS MANUAL
COMPUTER PROGRAM TEST REPORT
PROGRAM PACKAGE DOCUMENT

THE FOLLOWING DOCUMENTS MAY BE REQUIRED AT VARIOUS POINTS THROUGHOUT THE SOFTWARE LIFE CYCLE:

SOFTWARE CHANGE PROPOSAL
SOFTWARE ENHANCEMENT PROPOSAL
SOFTWARE TROUBLE REPORT

OK - ANSWER THE FOLLOWING QUESTIONS TO OBTAIN GUIDANCE SPECIFIC TO THE UNIQUE CHARACTERISTICS OF YOUR SYSTEM.

MULTIPLE COMPUTER PROGRAMS:
(1)YES
(2)NO
=?
display text(program description document).

COMMAND ENCOUNTERED-DEFERRING CURRENT QUESTION

THE PROGRAM DESCRIPTION DOCUMENT (PDD) PROVIDES A COMPLETE TECHNICAL DESCRIPTION OF ALL DIGITAL PROCESSOR SUBPROGRAM FUNCTIONS, STRUCTURES, OPERATION ENVIRONMENTS, OPERATING CONSTRAINTS, DATA BASE ORGANIZATION, SOURCE AND OBJECT CODE LISTING, AND DIAGRAMMATIC/NARRATIVE FLOWS. EACH PROGRAM DESCRIPTION DOCUMENT IS DIRECTLY RESPONSIBLE TO THE PROGRAM DESIGN SPECIFICATION AND TO ANY APPROPRIATE SOFTWARE AND/OR PROGRAM SPECIFICATION.

READY FOR COMMAND:

display text(tadstand-c).

THIS TADSTAND PROMULGATES POLICY FOR THE STANDARDIZATION OF COMPUTER PROGRAMMING LANGUAGES USED IN THE DEVELOPMENT, ACQUISITION, DEPLOYMENT, AND SUPPORT OF TACTICAL DIGITAL SYSTEMS. THE STANDARD LANGUAGES ARE:

STANDARD PROCESSOR	STANDARD LANGUAGES
AN/UYK-20, 44, & 14	CMS-2M, CMS-2Y(20), & SPL/I(M)
AN/UYK-7, & 43	CMS-2Y
AN/UYS-1	

READY FOR COMMAND:

continue.

CONTINUING PREVIOUS LINE OF QUESTIONING

```
MULTIPLE COMPUTER PROGRAMS:
     (1)YES
     (2)NO
=?
2.

MULTIPLE DEVELOPMENT AGENTS:
     (1)YES
     (2)NO
=?
2.

INFORMATION IS AVAILABLE FOR PPS:
     (1)YES
     (2)NO
=?
1.

ACQUISITION PHASE:
     (1)EXPLORATORY DEVELOPMENT OR ADVANCED DEVELOPMENT
     (2)ENGINEERING DEVELOPMENT
     (3)IN-SERVICE ENGINEERING
=?
3.

FIRMWARE:
     (1)YES
     (2)NO
=?
1.

PROCESSOR TYPE:
     (1)NAVY STANDARD
     (2)NON-STANDARD
=?
2.

SECURITY:
     (1)MULTI-LEVEL SECURITY
     (2)SECURE COMMUNICATIONS
     (3)NONE
=?
3.

INTERFACES TO OTHER SYSTEMS:
     (1)YES
     (2)NO
=?
1.
```

MULTIPLE COMPUTERS:
(1)YES
(2)NO
=?
2.

THE FOLLOWING LIST PROVIDES SPECIFIC GUIDANCE ON REQUIREMENTS RELATIVE TO THE UNIQUE CHARACTERISTICS OF YOUR SYSTEM:

EITHER TYPE B SPECIFICATION OF MIL-STD-490 OR SOD OF SECNAVINST 3560.1 IS RECOMMENDED. ONE PPS FOR EACH COMPUTER PROGRAM IS ALSO RECOMMENDED.

THE ONLY NAVY SPECIFIC STANDARD REQUIRED FOR FIRMWARE IS MIL-STD-1679 BUT QA/CM STANDARDS MUST BE REPLACED BY CORRESPONDING HARDWARE STANDARDS AFTER ACCEPTANCE.

NON-STANDARD PROCESSOR WAIVER IS REQUIRED ON TADSTAND B AND C. DODI-5000.31 BECOMES THE GUIDING DOCUMENT ON LANGUAGE SELECTION.

AN IDS IS REQUIRED FOR EACH INTERSYSTEM INTERFACE.

READY FOR COMMAND:

stop.

KES TERMINATED

B.3 Standards Knowledge Base

This section contains a listing of the entire standards demonstration knowledge base in source text form. This knowledge base represents 25% of what might be required for an operational software standards expert system. The standards demonstration knowledge base was designed, researched, and implemented in six man-weeks of effort.

```
ATTACHMENTS:
     TEXT
%
ATTRIBUTES:
     TYPE OF SYSTEM(SGL):
          ADP, TACTICAL, NOT SURE.
     CLASSIFICATION(SGL):
          ANTI-AIR WARFARE, ANTI-SUBMARINE WARFARE, ANTI-
          SURFACE WARFARE, STRIKE WARFARE, AMPHIBIOUS
          WARFARE, MINE WARFARE, SPECIAL WARFARE, MOBILITY,
          COMMAND CONTROL AND COMMUNICATIONS, INTELLIGENCE,
          ELECTRONIC WARFARE, NCO 2-6 '9 '18 '20, OTHER.
     OPERATIONAL BASE(SGL):
          AIR OR SUBMARINE OR SURFACE SHIP, SHORE.
     PART OF NCCS(SGL):
          YES, NO.
     MULTIPLE COMPUTER PROGRAMS(SGL):
          YES, NO.
     MULTIPLE DEVELOPMENT AGENTS(SGL):
          YES, NO.
     ACQUISITION PHASE(MLT):
          EXPLORATORY DEVELOPMENT OR ADVANCED DEVELOPMENT,
          ENGINEERING DEVELOPMENT, IN-SERVICE ENGINEERING.
     PROCESSOR TYPE(MLT):
          NAVY STANDARD, NON-STANDARD.
     SECURITY(MLT):
          MULTI-LEVEL SECURITY, SECURE COMMUNICATIONS, NONE.
     FIRMWARE(SGL):
          YES, NO.
     MULTIPLE COMPUTERS(SGL):
          YES, NO.
     INTERFACES TO OTHER SYSTEMS(SGL):
          YES, NO.
     GUIDELINES(MLT):
          DODD-5000-1 MAJOR SYSTEMS ACQUISITION
          [SYNONYMS:  DODD-5000-1]
          [TEXT:  "DEPARTMENT OF DEFENSE DIRECTIVE
          (DODD)5000-1 'MAJOR SYSTEM" "ACQUISITION'
          DEFINES THE DOD POLICIES FOR MAJOR SYSTEM"
          "PROGRAM ACQUISITION.  DODD-5000-1 DEFINES 4
          PHASES OF "PROGRAM ACTIVITY:"
          "  "
          "  PROGRAM INITIATION,"
          "  DEMONSTRATION AND VALIDATION,"
          "  FULL-SCALE ENGINEERING DEVELOPMENT, AND"
          "  PRODUCTION AND DEPLOYMENT."]
```

```
        DODI-5000-2 MAJOR SYSTEM ACQUISITION PROCESS
        [SYNONYMS:  DODI-5000-2]
        [TEXT:  "DEPARTMENT OF DEFENSE DIRECTIVE (DODD)
                5000-2 'MAJOR SYSTEM"
                "ACQUISITION PROCESS' SUPPLEMENTS DODD-
                5000-1 WITH POLICIES"
                "AND PROCEDURES.  THIS DIRECTIVE
                STRUCTURES THE SYSTEM"
                "ACQUISITION PROCESS SO THAT PROGRAMS
                PROGRESS THROUGH"
                "AN ESTABLISHED SET OF DECISION POINTS
                AND PHASES TO"
                "COMPLETION OR TERMINATION."],
        DODI-5000-29 MANAGEMENT OF COMPUTER RESOURCES
        IN MAJOR DEFENSE SYSTEMS
        [SYNONYMS:  DODD-5000-29]
        [TEXT: "    "
        ],
        DODI-5000-31 LANGUAGE STANDARDIZATION
        [SYNONYMS:  DODI-5000-31]
        [TEXT:  "DEPARTMENT OF DEFENSE INSTRUCTION
                (DODI) 5000-31 'INTERIM"
                "LIST OF HIGH ORDER PROGRAMMING LANGUAGES
                (HOL' SPECIFIES"
                "THE HIGH ORDER PROGRAMMING LANGUAGES
                APPROVED FOR USE.  THE"
                "APPROVED LANGUAGES ARE:"
                "      MCS-2 (NAVY)"
                "      SPL-1 (NAVY)"
                "      TAPCOL (ARMY)"
                "      JOVIAL (AIR FORCE)"
                "      COBOL"
                "      FORTRAN."],
        SECNAVINST-5200-32 MANAGEMENT OF COMPUTER
        RESOURCES IN MAJOR DEFENSE SYSTEMS
        [SYNONYMS:  SECNAVINST-5200-32],
        SECNAVINST-5231-1A,
           SECNAVINST-5230-4.
    TOP LEVEL REQUIREMENTS(MLT):
        DOD-STD-480A CONFIG CONTROL ENGINEERING CHANGES
        DEVIATIONS AND WAIVERS
        [SYNONYMS:  DOD-STD-480A]
        [TEXT:  "DOD-STD-480A 'CONFIGURATION CONTROL -
                ENGINEERING CHANGES"
                "DEVIATIONS AND WAIVERS (SHORT FORM)'
                DELINEATES CONFIGURATION"
                "CONTROL REQUIREMENTS AND PROVIDES
                INSTRUCTIONS FOR PREPARING"
                "AND SUBMITTING PROPOSED ENGINEERING
                CHANGES AND RELATED" INFORMATION.  THIS
                STANDARD IS INTENDED FOR USE IN CONTRACTS"
```

```
                "INVOLVING THE PROCUREMENT OF MULTI-
                APPLICATION ITEMS OR"
                "ITEMS FOR WHICH THE PRESCRIBED DETAILED
                DESIGN WAS NOT"
                "DEVELOPED BY THE CONTRACTOR.  WHERE A
                MORE COMPLETE"
                "DESCRIPTION OF ENGINEERING CHANGES IS
                DESIRED, DOD-STD-480A"
                "SHOULD BE SPECIFIED INSTEAD."],
        MIL-STD-490 SPECIFICATION PRACTICES
        [SYNONYMS: MIL-STD-490]
        [TEXT:  "MIL-STD-490 'SPECIFICATION PRACTICES'
                SETS FORTH PRACTICES"
                "FOR THE PREPARATION, INTERPRETATION,
                CHANGE, AND REVISION"
                "OF PROGRAM PECULIAR SPECIFICATIONS.
                MIL-STD-490 DEFINES "
                "THE FORMAT AND CONTENT OF THE FOLLOWING
                TYPES OF"
                "SPECIFICATIONS"
                "     TYPE A SYSTEM SPECIFICATION"
                "     TYPE B DEVELOPMENT SPECIFICATIONS"
                "     TYPE C PRODUCT SPECIFICATIONS"],
        SECNAVINST-5236-1B,
        SECNAVINST-5501-11,
        SECNAVINST-5239-1.
NAVY-SPECIFIC AND DETAILED REQUIREMENTS(MLT):
        SECNAVINST-3560-1 TACTICAL DIGITAL SYSTEMS
        DOCUMENTATION STANDARDS
        [SYNONYMS:  SECNAVINST-3560-1]
        [TEXT:  "SECNAVINST-3560-1 'TACTICAL DIGITAL
                SYSTEMS DOCUMENTATION"
                "STANDARDS' IDENTIFIES AND DESCRIBES
                DOCUMENTS NECESSARY TO"
                "SUPPORT THE DEVELOPMENT AND MAINTENANCE
                OF DIGITAL PROCESSOR"
                "PROGRAMS FOR TACTICAL SYSTEMS.
                SECNAVINST-3560-1 IS THE"
                "VEHICLE BY WHICH THE PROCURING AGENCY
                SELECTS THE NECESSARY"
                "DOCUMENTATION FOR PROJECT DEVELOPMENT."],
        NAVMATINST-4130-2A CONFIG MNGMT OF SOFTWARE FOR
        TACTICAL SYSTEMS
        [SYNONYMS:  NAVMATINST-5200-27A]
        NAVMATINST-5200-27A PROCS FOR TRANSFER OF NAVY
        TACTICAL SYSTLEM SW RESP
        [SYNONYMS:  NAVMATINST-5200-27A]
        [TEXT:  "NAVMAT INSTRUCTION 5200-27A'TRANSFER
                OF NAVY TACTICAL"
                "DIGITAL SYSTEM SOFTWARE RESPONSIBILITY:
                PROCEDURES FOR'"
```

```
        "PROMULGATES POLICY AND PROCEDURES FOR
        THE TRANSFER OF NAVY"
        "TACTICAL DIGITAL SYSTEM SOFTWARE
        RESPONSIBILITIES FROM A"
        "DEVELOPMENTAL ACTIVITY TO A PROGRAM
        MAINTENANCE ACTIVITY."],
TADSTAND-A STANDARD DEFINITIONS FOR ECR
IN TACTICAL DIGITAL SYSTEMS
[SYNONYMS:  TADSTAND-A]
[TEXT:  "TADSTAND-A 'STANDARD DEFINITIONS FOR
        EMBEDDED COMPUTER"
        "RESOURCES IN TACTICAL DIGITAL SYSTEMS'
        PROVIDES STANDARD"
        "DEFINITIONS FOR EMBEDDED COMPUTER
        RESOURCES (ECR).  THESE"
        "STANDARD DEFINITIONS APPLY TO TERMS
        USED IN ALL OTHER"
        "TADSTANDS"],
TADSTAND-B STANDARD EMBEDDED COMPUTERS AND
PERIPHERALS
[SYNONYMS:  TADSTAND-B]
[TEXT:  "TADSTAND-B 'STANDARD EMBEDDED COMPUTERS,
        COMPUTER PERIPHERALS"
        "AND INPUT/OUTPUT INTERFACES' IDENTIFIES
        SPECIFIC EQUIPMENT"
        "MODELS AND I/O INTERFACES AS STANDARDS
        OR AS PLANNED"
        "STANDARDS"],
TADSTAND-C LANGUAGE STANDARDIZATION POLICY
FOR TACTICAL SYSTEMS
[SYNONYMS:  TADSTAND-C]
[TEXT:  "THIS TADSTAND PROMULGATES POLICY FOR
        THE STANDARDIZATION OF"
        "COMPUTER PROGRAMMING LANGUAGES USED IN
        THE DEVELOPMENT,"
        "ACQUISITION, DEPLOYMENT, AND SUPPORT
        OF TACTICAL DIGITAL"
        "SYSTEMS.  THE STANDARD LANGUAGES ARE:"
        "  "
        "     STANDARD PROCESSOR   STANDARD LANGUAGES"
        "     ------------------   ------------------"
        "     AN/UYK-20, 44, & 14  CMS-2M, CMS-2Y(20),"
        "  "                         & SPL/I(M)"
        "     AN/UYK-7, & 43       CMS-2Y"
        "  "
        "     AN/UYSL-1            SPL/I"],
TADSTAND-D RESERVE CAPACITY REQUIREMENTS FOR TACTICAL
DIGITAL SYSTEMS
[SYNONYMS:  TADSTAND-D]
[TEXT:  "TADSTAND-D 'RESERVE CAPACITY REQUIREMENTS
        FOR TACTICAL"
```

```
        "DIGITAL SYSTEMS' SPECIFIES THE FOLLOWING
        RESERVE CAPACITIES:"
        "     MAIN MEMORY                20%"
        "     SECONDARY STORAGE          20%"
        "     CPU THROUGHPUT             20%"
        "     NUMBER OF I/O CHANNELS     18.75%"
        "     I/O CHANNEL THROUGHPUT     20%"],
TADSTAND-2 STANDARD SPECIFICATION FOR TACTICAL
PROGRAM DOCUMENTATION
[SYNONYMS:  TADSTAND-2]
[TEXT:  "TADSTAND-2 'STANDARD SPECIFICATION FOR
        TACTICAL DIGITAL"
        "COMPUTER PROGRAM DOCUMENTATION'
        REFERENCES SECNAVINST-3560-1"
        "PARAGRAPHS 5 AND 6 AS A STANDARD"],
TADSTAND-3 REQUIREMENTS FOR INTER-DIGITAL
PROCESSOR INTERFACE DOC
[SYNONYMS:  TADSTAND-3]
[TEXT:  "TADSTAND-3 'STANDARD REQUIREMENTS FOR
        INTER-DIGITAL"
        "PROCESSOR INTERFACE DOCUMENTATION'
        REFERENCES PARAGRAPHS"
        "5 AND 6 OF SECNAVINST-3560-1 AS A
        STANDARD.  THIS STANDARD"
        "CALLS FOR THE INTERFACE DESIGN
        SPECIFICATION (IDS) TO"
        "SATISFY THIS REQUIREMENT"],
TADSTAND-9 SW QUALITY TESTING CRITERIA STANDARD
FOR TACTICAL SYSTEMS
[SYNONYMS:  TADSTAND-9]
[TEXT:  "TADSTAND-9 'SOFTWARE QUALITY TESTING
        CRITERIA STANDARD"
        "FOR TACTICAL DIGITAL SYSTEMS' PROVIDES
        SPECIFIC"
        "DEFINITIONS, REQUIREMENTS, AND
        LIMITATIONS RELEVANT TO"
        "COMPUTER PROGRAM TESTING, ERROR
        RECORDING AND PATCHING."],
MIL-STD-1679 MILITARY STANDARD FOR WEAPON
SYSTEM SOFTWARE DEVELOPMENT
[SYNONYMS:  MIL-STD-1679]
[TEXT:  "THIS STANDARD ESTABLISHES UNIFORM"
        "REQUIREMENTS FOR THE DEVELOPMENT OF"
        "WEAPON SYSTEM SOFTWARE WITHIN THE"
        "DEPARTMENT OF DEFENSE.  WHEN INVOKED IN"
        "A SPECIFICATION OR STATEMENT OF WORK,"
        "THESE REQUIREMENTS SHALL APPLY TO THE"
        "WEAPON SYSTEM SOFTWARE (INCLUDING"
        "FIRMWARE) WHICH IS DEVELOPED EITHER"
        "ALONE OR AS A PORTION OF A WEAPON SYSTEM"
        "OR SUBSYSTEM.
```

```
                "THIS STANDARD REFERENCES DOD-STD-480A,"
                "MIL-STD-481, NAVMAT 'TACTICAL DATA"
                SYSTEMS GLOSSARY', ANSI 3.12-1970,"
                AND NAVSO ADP GLOSSARY."],
        MIL-S-52779A SOFTWARE QUALITY ASSURANCE PROGRAM
        REQUIREMENT
        [SYNONYMS:  MIL-S-52779A]
        [TEXT:  "MIL-S-52779A 'SOFTWARE QUALITY ASSURANCE"
                "PROGRAM REQUIREMENTS' APPLIES TO"
                "ACQUISITION OF SOFTWARE WHERE THE"
                "ACQUISITION INVOLVES EITHER SOFTWARE"
                "ALONE OR SOFTWARE AS A PORTION OF A"
                "SYSTEM OR SUBSYSTEM.  THIS SPECIFICATION"
                "REQUIRES THE ESTABLISHMENT OF AND IMPLE-
                "MENTATION OF A SOFTWARE QUALITY"
                "ASSURANCE (SQA) PROGRAM BY THE CONTRACTOR."],
        MIL-STD-1521A TECHNICAL REVIEWS AND AUDITS
        [SYNONYMS:  MIL-STD-1521A]
        [TEXT:  "MIL-STD-1521A 'TECHNICAL REVIEWS AND"
                AUDITS FOR SYSTEMS, EQUIPMENTS, AND"
                "COMPUTER PROGRAMS' PRESCRIBED THE REQUIRE-"
                "MENTS FOR THE CONDUCT OF TECHNICAL "
                "REVIEWS AND AUDITS.  THE TYPE OF TECH-"
                "NICAL REVIEWS AND AUDITS THAT MAY BE"
                "SELECTED INCLUDE, BUT ARE NOT LIMITED"
                "TO, THE FOLLOWING:"
                "     SYSTEM REQUIREMENTS REVIEW (SRR)"
                "     SYSTEM DESIGN REVIEW (SDR)"
                "     PRELIMINARY DESIGN REVIEW (PDR)"
                "     CRITICAL DESIGN REVIEW (CDR)"
                "     FUNCTIONAL CONFIGURATION AUDIT (FCA)"
                "     PHYSICAL CONFIGURATION AUDIT (PCA)"
                "     FORMAL QUALIFICATION REVIEW (FQR)"
        SECNAVINST-5233-1B,
        SECNAVINST-5211-5B.
TOP LEVEL SPECIFICATION REQUIREMENT(VAL).
SYSTEM TYPE(SGL):
        ADP, TACTICAL.
MANDATORY REQUIREMENTS AREAS OF CONCERN(SGL):
        APPLICATION SOFTWARE DOCUMENTATION
        [TEXT:  "THE FOLLOWING LIST OF DOCUMENTS ARE"
                "MANDATORY FOR TACTICAL DIGITAL APPLICATION"
                "SOFTWARE.  EACH DOCUMENT TITLE IS"
                "GROUPED WITH OTHER DOCUMENTS THAT ARE"
                "ACCEPTED AT APPROXIMATELY THE SAME"
                "TIME.  NOTE THAT BEFORE ANY DOCUMENT"
                "IN A GROUP CAN BE ACCEPTED ALL OF THE"
                "DOCUMENTS IN THE PRECEEDING GROUP MUST
                "BE ACCEPTED."
                "     "
                "1.  TYPE B SPEC OR SOD OR FOD"
                "     "
```

```
                "2.  SOFTWARE DEVELOPMENT PLAN"
                "    SOFTWARE QUALITY ASSURANCE PLAN"
                "    SOFTWARE CONFIGURATION MANAGEMENT"
                       PLAN"
                "    "
                "3.  PROGRAM PERFORMANCE SPECIFICATION"
                "    INTERFACE DESIGN SPECIFICATION"
                "    COMPUTER PROGRAM TEST PLAN"
                "    COMPUTER PROGRAM TEST SPECIFICATION"
                "    "
                "4.  PROGRAM DESIGN SPECIFICATION"
                "    COMPUTER PROGRAM TEST PROCEDURES"
                "    "
                "5.  PROGRAM DESCRIPTION DOCUMENT"
                "    DATA BASE DESIGN DOCUMENT"
                "    OPERATORS MANUAL"
                "    SYSTEM OPERATORS MANUAL"
                "    COMPUTER PROGRAM TEST REPORT"
                "    PROGRAM PACKAGE DOCUMENT"
                "    "
                "THE FOLLOWING DOCUMENTS MAY BE REQUIRED"
                "AT VARIOUS POINTS THROUGHOUT THE SOFTWARE"
                "LIFE CYCLE:"
                "    "
                "    SOFTWARE CHANGE PROPOSAL"
                "    SOFTWARE ENHANCEMENT PROPOSAL"
                "    SOFTWARE TROUBLE REPORT"
                "    "],
        HARDWARE, SUPPORT SOFTWARE.
    STANDARDS IMPLEMENTATION GUIDANCE [SYNONYMS:  SIG] (MLT):
        TYPE-B-OR-SOD-FOD
        [TEXT:  "EITHER TYPE B SPECIFICATION OF MIL-STD-490"
                "OR SOD OF SECNAVINST 3560.1 IS RECOMMENDED."
                "ONE PPS FOR EACH COMPUTER PROGRAM IS ALSO"
                "RECOMMENDED."
                ],
        PPS-ALONE
        [TEXT:  "PPS ALONE IS SUFFICIENT FOR TOP LEVEL"
                "SPECIFICATION."
                ],
        RELAX
        [TEXT:  "THE FOLLOWING MAY BE RELAXED FOR ADVANCED"
                "OR EXPLORATORY DEVELOPMENT:
                "    - MIL-STD-1521A  AUDITS
                "    - MIL-STD-1679   STATUS REVIEWS
                                      (5.12.3)"
                "                     INSPECTIONS AND
                                      AUDITS (5.12.4)"
                                      QUALITY ASSURANCE
                                      (5.10.2)"
                                      TESTING AT ALL LEVELS
                                      (5.8)"
```

```
                    - TADSTAND 9     QUALITY ASSURANCE
                ],
        FIRMWARE
        [TEXT:  "THE ONLY NAVY SPECIFIC STANDARD REQUIRED"
                "FOR FIRMWARE IS MIL-STD-1679 BUT QA/CM"
                "STANDARDS MUST BE REPLACED BY CORRESPOND-"
                "ING HARDWARE STANDARDS AFTER ACCEPTANCE."
                ],
        PROCESSOR-WAIVER
        [TEXT:  "NON-STANDARD PROCESSOR WAIVER IS REQUIRED"
                "ON TADSTAND B AND C.  DODI 5000.31 BECOMES"
                "THE GUIDING DOCUMENT ON LANGUAGE"
                "SELECTION."],
        SECURE
        [TEXT:  "FOR SECURITY ENHANCEMENTS THE FOLLOWING"
                "ADDITIONS TO THE NAVY SPECIFIC STANDARDS"
                "SHOULD BE CONSIDERED:
                "     - JCP PUB22 WWMCCS      ADP SYSTEM"
                "                             SECURITY MANUAL"
                "     - OPNAVINST 5510.1      DEPT OF NAVY"
                "                             SECURITIY"
                "                             REGULATION"
                "     - OPNAVINST 5510.45     U.S. NAVY"
                "                             PHYSICAL"
                "                             SECURITY MANUAL"
                "     - NAVELEXINST 5510.2    GUIDELINES FOR"
                "                             COMPROMISING"
                "                             EMANATIONS"
                "                             CONTROL."
                ],
        INTERSYSTEM
        [TEXT:  "AN IDS IS REQUIRED FOR EACH INTERSYSTEM"
                "INTERFACE."],
        NO-INTERSYSTEM
        [TEXT:  "THERE IS NO REQUIREMENT FOR IDS'S FOR"
                "YOUR SYSTEM."],
        INTERCOMPUTER
        [TEXT:  "IDS'S ARE RECOMMENDED FOR INTERCOMPUTER"
                "INTERFACES, BUT ARE NOT MANDATORY."],
THE FOLLOWING ATTRIBUTES ARE NOT INCLUDED IN THE ATTRIBUTE
HIERARCHY (IE:  THEY ARE NOT REFERENCED IN ANY RULES).
THEY ARE USED SOLELY AS POINTERS TO THE ASSOCIATED FREE
TEXT.

    TYPE-B
        [TEXT:  "THE TYPE-B SPECIFICATION DESCRIBES IN"
                "OPERATIONAL, FUNCTIONAL AND MATHEMATICAL"
                "LANGUAGE ALL OF THE REQUIREMENTS"
                "NECESSARY TO DESIGN AND VERIFY THE"
                "COMPUTER PROGRAM IN TERMS OF PERFORMANCE"
                "CRITERIA.  TYPE-B SPECIFICATIONS ARE"
                "DEFINED IN MIL-STD-490."] (VAL).
```

```
SYSTEM OPERATIONAL DESIGN [SYNONYMS:  SOD]
     [TEXT:  "THE SYSTEM OPERATIONAL DESIGN (SOD) IS"
             "A TECHNICAL PLANNING DOCUMENT THAT DEFINES"
             "THE ENVIRONMENT (HARDWARE), PROGRAM"
             "FUNCTIONAL OPERATIONS (SOFTWARE) AND THE"
             "DECISION MAKING INTERFACE BETWEEN THEM"
             "FOR IMPLEMENTING THE PROGRAM.  THE"
             "SOD CONTAINS ALL PROPOSED PROGRAM"
             "FUNCTION CORE ALLOCATIONS, ALL SUBPROGRAM"
             "DEFINITIONS AND THEIR INTERFACES,"
             OVERALL" PROGRAM DATA STORAGE PLANS,"
             "AND ANY SUPPORT PROGRAMS THAT WILL BE"
             "REQUIRED FOR FURTHER DEVELOPMENT.  THE"
             SOD IS DEFINED IN SECNAVINST-3560-1."
     ] (VAL).
FUNCTION OPERATIONAL DESIGN [SYNONYMS:  FOD]
     [TEXT:  "THE FUNCTION OPERATIONAL DESIGN (FOD)"
             DESCRIBED IN DETAIL THE OPERATOR OR"
             "OR EQUIPMENT ACTION, DATA SOURCES,"
             "CONTROL REQUIREMENTS AND CONTROL DESIGN,"
             "AND THEIR INTERFACES FOR EACH CONSOLE"
             "AND MODE OF OPERATION.  THE FOD IS"
             "DEFINED IN SECNAVINST-3560-1."] (VAL).
SOFTWARE DEVELOPMENT PLAN [SYNONYMS:  SDP]
     [TEXT:  "THE SOFTWARE DEVELOPMENT PLAN DESCRIBED"
             "THE COMPREHENSIVE PLAN FOR THE MANAGEMENT"
             "OF THE DEVELOPMENT EFFORT FOR THE COMPUTER"
             "PROGRAM.  THE SDP INCLUDES A DESCRIPTION"
             "OF THE DEVELOPMENT ORGANIZATION, A"
             "DESCRIPTION OF THE DESIGN" APPROACH,"
             "MILESTONES AND SCHEDULES, AND RESOURCE"
             "ALLOCATION."] (VAL).
SOFTWARE QUALITY ASSURANCE PLAN
     [TEXT:  "THE SOFTWARE QUALITY ASSURANCE (QA)"
             "PLAN DESCRIBED THE ORGANIZATION AND"
             "PROCEDURES USED TO ASSURE THAT SOFTWARE"
             "COMPLIES WITH THE SPECIFICATION.  THE"
             "PLAN IS ORIENTED TOWARD THE DESIGN AND"
             "PRODUCTION OF SOFTWARE THAT IS EFFECTIVE"
             "AND RELIABLE, AND THAT IS PLANNED AND"
             "DEVELOPED IN CONSONANCE WITH OTHER"
             "ADMINISTRATIVE AND TECHNICAL PROGRAM"
             ] (VAL).
SOFTWARE CONFIGURATION MANAGEMENT PLAN
     [SYNONYMS:  SCMP]
     [TEXT:  "THE 'SOFTWARE CONFIGURATION MANAGEMENT"
             "PLAN' (SCMP) DESCRIBES THE ORGANIZATION"
             "AND PROCEDURES USED TO CONTROL SOFTWARE"
             "CONFIGURATION ITEMS.  THE PLAN DESCRIBES"
             "PROCEDURES FOR THE IDENTIFICATION, CONTROL,"
             "AUTHENICATION, AND STATUS ACCOUNTING"
```

```
                "OF SOFTWARE CONFIGURATION ITEMS.  IN"
                "ADDITION, THE PLAN DESCRIBES THE CON-"
                "FIGURATION MANAGEMENT ORGANIZATION"
                "AND THE PROCEDURES FOR ENSURING TOTAL"
                "SYSTEM COMPATABILITY."] (VAL).
PROGRAM PERFORMANCE SPECIFICATION
     [SYNONYMS:  PPS]
     [TEXT:  "THE PROGRAM PERFORMANCE SPECIFICATION"
             "(PPS) DESCRIBES IN DETAIL ALL THE"
             "OPERATIONAL AND FUNCTIONAL REQUIREMENTS"
             "NECESSARY TO DESIGN, TEST, AND MAINTAIN"
             "THE REQUIRED DIGITAL PROCESSOR PROGRAMS."
             "IT PROVIDES THE LOGICAL, DETAILED"
             "DESCRIPTIONS OF THE PERFORMANCE REQUIRE-"
             "MENTS OF A DIGITAL PROCESSOR PROGRAM."
     ] (VAL),.
INTERFACE DESIGN SPECIFICATION
     [SYNONYMS:  IDS]
     [TEXT:  "THE INTERFACE DESIGN SPECIFICATION (IDS)"
             "ESTABLISHES A SET OF REQUIREMENTS FOR"
             "THE DESIGN DETERMINATIONS OF THE INTER-"
             "DIGITAL PROCESSOR DIGITAL INTERFACES."
             "IT PROVIDES A DETAILED LOGICAL DESCRIPTION"
             "OF:  ALL DATA UNITS, ALL MESSAGES, USE"
             "OF ALL CONTROL SIGNALS FOR DEFINING"
             INTER-DIGITAL PROCESSOR COMMUNICATIONS"
             "CONVENTIONS."] (VAL).
COMPUTER PROGRAM TEST PLAN
     [TEXT:  "THE COMPUTER PROGRAM TEST PLAN DEFINES"
             "THE TOTAL SCOPE OF THE TESTING TO BE"
             "PERFORMED.  IT IDENTIFIES THE PARTICULAR"
             "LEVEL OF TESTING AND DESCRIBES ITS"
             "CONTRIBUTING ROLE FOR ENSURING THE"
             "RELIABILITY AND CERTIFIED ACCEPTANCE OF"
             "THE COMPUTER PROGRAM.  INDIVIDUAL TEST"
             "REQUIREMENTS ARE LISTED FOR EVERY TEST"
             "TO BE CONDUCTED.  THE TEST PLAN CONTAINS"
             "PRECISE STATEMENTS OF THE PURPOSE, SCOPE,"
             "AND SCHEDULE FOR EACH INDIVIDUAL TEST."
             "IT IDENTIFIES THE DEGREE OF TESTING"
             "AND THE SPECIFIC FUNCTIONS INVOLVED IN"
             "THE TEST.  ALSO, THE SPECIFIC OBJECTIVES"
             "OF THE TEST ARE DEFINED AND A SUMMARY"
             "OF THE TEST METHODS AND THE TYPE OF"
             "SYSTEM ENVIRONMENT TO BE USED ARE"
             "INCLUDED."] (VAL.)
COMPUTER PROGRAM TEST SPECIFICATION
     [TEXT:  "THE COMPUTER PROGRAM TEST SPECIFICATION"
             "IS PREPARED FOR EACH TEST SPECIFIED IN"
             "THE CORRESPONDING TEST PLAN; NORMALLY"
             "ONE FOR EACH SUBPROGRAM OR SPECIFIED"
```

```
                "FUNCTION AND ONE FOR THE PERTINENT TEST."
                "IT IDENTIFIES THE BASIC TEST CRITERIA"
                "AND METHOD OF TESTING.  THE TEST SPECI-"
                "FICATION IS DERIVED FROM THE CORRESPONDING"
                "SYSTEM OR PROGRAM SPECIFICATION."] (VAL).
PROGRAM DESIGN SPECIFICATION [SYNONYMS:  PDS]
     [TEXT:  "THE PROGRAM DESIGN SPECIFICATION (PDS)"
             "IS THE DESIGN DESCRIPTION OF THE DIGITAL"
             "PROCESSOR PROGRAM.  IT IS BASED UPON"
             "THE PERFORMANCE REQUIREMENTS DEFINED"
             "IN THE PROGRAM PERFORMANCE SPECIFICATION"
             "(PPS).  THE INTERFACE DESIGN SPECIFI-"
             "CATION (IDS) IS ALSO USED AS INPUT TO"
             "THIS SPECIFICATION."] (VAL).
COMPUTER PROGRAM TEST PROCEDURES
     [TEXT:  "THE COMPUTER PROGRAM TEST PROCEDURES"
             "PROVIDE DETAILED INSTRUCTIONS FOR"
             "TEST EXECUTION AND FOR EVALUATION"
             "OF THE RESULTS FOR EACH LEVEL OF TESTING"
             "SPECIFIED.  THE TEST PROCEDURES ARE"
             "DEVELOPED FROM THE TEST SPECIFICATION"
             "AND RELEVANT DESIGN DOCUMENTS.  THEY"
             "PRESENT DETAILED INSTRUCTIONS FOR TEST"
             "SETUP, EXECUTION, AND THE EVALUATION"
             "OF TEST RESULTS."] (VAL).
PROGRAM DESCRIPTION DOCUMENT [SYNONYMS:  PDD]
     [TEXT:  "THE PROGRAM DESCRIPTION DOCUMENT (PDD)"
             "PROVIDES A COMPLETE TECHNICAL DESCRIP-"
             "TION OF ALL DIGITAL PROCESSOR SUBPROGRAM"
             "FUNCTIONS, STRUCTURES, OPERATION"
             "ENVIRONMENTS, OPERATING CONSTRAINTS,"
             "DATA BASE ORGANIZATION, SOURCE AND"
             "OBJECT CODE LISTING, AND DIAGRAMMATIC/"
             "NARRATIVE FLOWS.  EACH PROGRAM DESCRIP-"
             "TION DOCUMENT IS DIRECTLY RESPONSIVE TO"
             "THE PROGRAM DESIGN SPECIFICATION AND TO"
             "ANY APPROPRIATE SOFTWARE AND/OR PROGRAM"
             "SPECIFICATION."] (VAL).
DATA BASE DESIGN DOCUMENT [SYNONYMS:  DBD]
     [TEXT:  "THE DATA BASE DESIGN DOCUMENT (DBD)"
             "PROVIDES A COMPLETE DETAILED DESCRIPTION"
             "OF ALL COMMON DATA ITEMS NECESSARY TO"
             "CARRY OUT THE FUNCTIONS OF THE DIGITAL"
             "PROCESSOR PROGRAM.  COMMON DATA IS"
             "THAT DATA REQUIRED BY TWO OR MORE SUB-"
             "PROGRAMS.  COMMON DATA INCLUDES CONSTANTS,
             "INDEXES, FLAGS, VARIABLES, AND TABLES."
             ] (VAL).
OPERATOR'S MANUAL [SYNONYMS:  OPERATORS MANUAL, OM]
     [TEXT:  "THE OPERATOR'S MANUAL (OM) PRESENTS"
             "PROCEDURES FOR PRESTANDBY/OPERATE,"
```

```
                    "MONITORING, AND RECOVERY OF THE DIGITAL"
                    "PROCESSOR PROGRAM.  IT IS LIMITED TO"
                    "INSTRUCTIONS FOR PREPARING AND MAINTAIN-"
                    "ING THE DIGITAL PROCESSOR PROGRAM IN"
                    "THE REQUIRED STATE OF CAPABILITY IN"
                    "ORDER THAT THE OPERATIONAL MISSION MAY"
                    "BE ACCOMPLISHED.  THE OPERATOR'S"
                    "MANUAL DOES NOT INCLUDE PROCEDURES FOR"
                    "SYSTEM OPERATION DIRECTLY IN SUPPORT"
                    "OF THE OPERATIONAL MISSION (THOSE PRO-"
                    "CEDURES ARE CONTAINED IN THE SYSTEM"
                    "OPERATOR'S MANUAL)."] (VAL).
     SYSTEM OPERATOR"S MANUAL [SYNONYMS:  SYSTEM OPERATORS
     MANUAL, SOM]
          [TEXT:    "THE SYSTEM OPERATOR'S MANUAL (SOM) IS"
                    "INTENDED TO BE THE SOLE REFERENCE"
                    "REQUIRED FOR INDIVIDUAL OPERATOR AND"
                    "STATION FUNCTION.  THE MANUAL IS USED"
                    "BY COMMAND OR SUPERVISORY PERSONNEL"
                    "FOR TRAINING OF OPERATORS.  IT ALSO"
                    "SERVES AS THE PRIME DOCUMENT FOR"
                    "OPERATIONAL NEEDS BY OPERATING "
                    "PERSONNEL.] (VAL).
     COMPUTER PROGRAM TEST REPORT
          [TEXT:    "COMPUTER PROGRAM TEST REPORTS DOCUMENT"
                    "THE RESULTS OF TESTS.  TEST REPORTS"
                    "DESCRIBE, DEFINE, AND EVALUATE DISCREP-"
                    "ANCIES BETWEEN THE DESIGN AND THE"
                    "ACTUAL PROGRAM.  THE REPORT DETAILS"
                    "ANY DEVIATIONS FROM THE TEST SPECIFIED"
                    "OR TEST PROCEDURES (EG:  SUBSTITUTION"
                    "OF EQUIPMENT, PROGRAM PATCHES, OR CHANGES"
                    "TO SUPPORT PROGRAMS) REQUIRED IN THE"
                    "COMPLETE PERFORMANCE OF THE TEST."] (VAL).
     PROGRAM PACKAGE DOCUMENT
          [TEXT:    "THE PROGRAM PACKAGE DOCUMENT CONSISTS OF"
                    "THE DIGITAL PROCESSOR SOURCE LISTING,"
                    "AN ERROR FREE SOURCE/OBJECT LISTING"
                    "PRODUCED BY AN ASSEMBLY OR COMPILATION"
                    "OF THE SOURCE CODE, AND ANY DATA WHICH"
                    "ARE NECESSARY TO CAUSE PROGRAMS TO"
                    "RUN PROPERLY (EG:  ADAPTATION DATA,"
                    "DATA FILE CONTENTS, SETUP DATA, OR"
                    "PROGRAM PARAMETER VALUES)."] (VAL).

%
RULES:
     TYPE1 IF
          TYPE OF SYSTEM = TACTICAL
           THEN
          SYSTEM TYPE = TACTICAL.
```

```
TYPE2 IF
      TYPE OF SYSTEM = ADP
       THEN
      SYSTEM TYPE = ADP.
TYPE3 IF
      TYPE OF SYSTEM = NOT SURE,
 & OPERATIONAL BASE = SHORE,
  & PART OF NCCS = NO
       THEN
      SYSTEM TYPE = ADP.
TYPE4 IF
      TYPE OF SYSTEM = NOT SURE,
 & OPERATIONAL BASE = SHORE,
  & PART OF NCCS = YES
       THEN
      SYSTEM TYPE = TACTICAL.
TYPE5 IF
      TYPE OF SYSTEM = NOT SURE,
 & OPERATIONAL BASE = SHORE,
  & PART OF NCCS = YES
       THEN
      SYSTEM TYPE = TACTICAL.
TYPE6 IF
      TYPE OF SYSTEM = NOT SURE,
 & OPERATIONAL BASE = AIR OR SUBMARINE OR SURFACE SHIP,
 & CLASSIFICATION = OTHER
       THEN
      SYSTEM TYPE = ADP.
SPEC1 IF
      MULTIPLE COMPUTER PROGRAMS = YES,
/     MULTIPLE DEVELOPMENT AGENTS = YES,
/     INFORMATION IS AVAILABLE FOR PPS = YES
       THEN
      TOP LEVEL SPECIFICATION REQUIREMENT.
GUIDELINE1 IF
      SYSTEM TYPE = TACTICAL
       THEN"
      GUIDELINES =
         DODD-5000-1
&        DODI-5000-2
&        DODD-5000-29
&        DODI-5000-31.
GUIDELINE2 IF
      SYSTEM TYPE = ADP
       THEN
      GUIDELINES =
         SECNAVINST-5231-1A
&        SECNAVINST-5230-4.
TOP1 IF
      SYSTEM TYPE = TACTICAL
       THEN
```

```
      TOP LEVEL REQUIREMENTS =
         DOD-STD-480A
&        MIL-STD-481A
&        MIL-STD-490.
TOP2 IF
      SYSTEM TYPE = ADP
       THEN
      TOP LEVEL REQUIREMENTS =
         SECNAVINST-5236-1B
&        SECNAVINST-5501-11
&        SECNAVINST-5239-1.
SPECIFIC1 IF
      SYSTEM TYPE = TACTICAL
       THEN
      NAVY-SPECIFIC AND DETAILED REQUIREMENTS =
         SECNAVINST-3560-1
&        NAVMATINST-4130-2A
&        NAVMATINST-5200-27A
&        TADSTAND-A & TADSTAND-B & TADSTAND-C
&        TADSTAND-D & TADSTAND-2 & TADSTAND-3 & TADSTAND-9
&        MIL-STD-1679
&        MIL-S-52779A
&        MIL-STD-1521A.
SPECIFIC2 IF
      SYSTEM TYPE = ADP
       THEN
      NAVY-SPECIFIC AND DETAILED REQUIREMENTS =
         SECNAVINST-5233-1B
&        SECNAVINST-5211-5B.
GUIDANCE1 IF
      SYSTEM TYPE = TACTICAL,
&     TOP LEVEL SPECIFICATION REQUIREMENT
       THEN
      SIG = TYPE-B-OR-SOD-FOD.
GUIDANCE3 IF
      SYSTEM TYPE = TACTICAL,
&     TOP LEVEL SPECIFICATION REQUIREMENT = ABSENT
       THEN
      SIG = PPS-ALONE.
GUIDANCE4 IF
      SYSTEM TYPE = TACTICAL,
&     ACQUISITION PHASE = EXPLORATORY DEVELOPMENT OR
      ADVANCED DEVELOPMENT
       THEN
      SIG = RELAX.
GUIDANCE5 IF
      SYSTEM TYPE = TACTICAL,
&     FIRMWARE = YES
       THEN
      SIG = FIRMWARE.
```

```
      GUIDANCE6 IF
           SYSTEM TYPE = TACTICAL,
      &    PROCESSOR TYPE = NON-STANDARD
            THEN
           SIG = PROCESSOR-WAIVER.
      GUIDANCE7 IF
           SYSTEM TYPE = TACTICAL,
      &    SECURITY = NONE
            THEN
           SIG = SECURE.
      GUIDANCE8 IF
           SYSTEM TYPE = TACTICAL,
      &    INTERFACES TO OTHER SYSTEMS = YES
            THEN
           SIG = INTERSYSTEM.
      GUIDANCE9 IF
           SYSTEM TYPE = TACTICAL,
      &    INTERFACES TO OTHER SYSTEMS = NO,
      &    MULTIPLE COMPUTERS = NO
            THEN
           SIG = NO-INTERSYSTEM.
      GUIDANCE10 IF
           SYSTEM TYPE = TACTICAL,
      &    MULTIPLE COMPUTERS = YES
            THEN
           SIG = INTERCOMPUTER
%
ACTIONS:
      MESSAGE "THIS IS A DEMONSTRATION KNOWLEDGE BASE ON"
              "THE SUBJECT OF SOFTWARE STANDARDS FOR DIGITAL"
              "SYSTEMS.  IT IS INTENDED FOR USE BY NAVY"
              "SOFTWARE ACQUISITION MANAGERS."
              "     "
              "THE KNOWLEDGE BASE IS ORGANIZED SUCH THAT"
              "STANDARDS REQUIREMENTS DEFINITION WILL PROCEED"
              "IN THREE STAGES IN THE FOLLOWING MANNER:"
              "     "
              "     STAGE 1 - DETERMINE GUIDELINES AND TOP"
              "               REQUIREMENTS AND NAVY-SPECIFIC"
              "               REQUIREMENTS WHICH MIGHT APPLY"
              "               TO YOUR SYSTEM."
              "     "
              "     STAGE 2 - DETERMINE MANDATORY REQUIRE-"
              "               MENTS FOR YOUR SYSTEM IN"
              "               THE AREA OF DOCUMENTATION,"
              "               HARDWARE, OR SUPPORT SOFTWARE."
              "     "
              "     STAGE 3 - DETERMINE GUIDANCE SPECIFIC"
              "               TO THE UNIQUE ASPECTS OF YOUR"
              "               SYSTEM."
              "     "
```

```
          "OK - FIRST ANSWER THE FOLLOWING QUESTIONS
          "TO DETERMINE WHICH STANDARDS APPLY TO YOUR"
          "SYSTEM.".
OBTAIN SYSTEM TYPE.
IF TYPE OF SYSTEM = NOT SURE THEN
     MESSAGE "THE TYPE OF SYSTEM HAS BEEN DETERMINED TO BE:",
     DISPLAY VALUE(SYSTEM TYPE),
     MESSAGE"    ".
OBTAIN GUIDELINES.
OBTAIN TOP LEVEL REQUIREMENTS.
OBTAIN NAVY-SPECIFIC AND DETAILED REQUIREMENTS.
MESSAGE "THE FOLLOWING LISTS OF GUIDELINES, TOP LEVEL"
     "REQUIREMENTS, AND NAVY SPECIFIC AND DETAILED REQUIRE-"
     "MENTS ARE APPLICABLE TO YOUR SYSTEM:".
MESSAGE "   "
     "***GUIDELINES***".
DISPLAY VALUE(GUIDELINES).
MESSAGE "   "
     "***TOP LEVEL REQUIREMENTS***:.
DISPLAY VALUE(TOP LEVEL REQUIREMENTS).
MESSAGE "   "
     "***NAVY SPECIFIC AND DETAILED REQUIREMENTS***:.
DISPLAY VALUE(NAVY-SPECIFIC AND DETAILED REQUIREMENTS).
IF SYSTEM TYPE = ADP THEN
     MESSAGE "   "
          "THIS SECTION OF THE KNOWLEDGE BASE HAS NOT"
          "YET BEEN IMPLEMENTED.",
     CLEAR.
MESSAGE "   "
     "OK - LET'S LOOK AT THE MANDATORY REQUIREMENTS FOR"
     "YOUR SYSTEM".
ASKFOR MANDATORY REQUIREMENTS AREAS OF CONCERN.
IF MANDATORY REQUIREMENTS AREAS OF CONCERN = HARDWARE/
SUPPORT SOFTWARE THEN
     MESSAGE "   "
          "THIS SECTION OF THE KNOWLEDGE BASE HAS NOT"
          "YET BEEN IMPLEMENTED.",
     CLEAR.
IF MANDATORY REQUIREMENTS AREAS OF CONCERN =
     APPLICATION SOFTWARE DOCUMENTATION THEN DISPLAY TEXT
     (APPLICATION SOFTWARE DOCUMENTATION).
MESSAGE "   "
     "OK - ANSWER THE FOLLOWING QUESTIONS TO OBTAIN"
     "GUIDANCE" SPECIFIC TO THE UNIQUE CHARACTERISTICS"
     "OF YOUR SYSTEM.".
OBTAIN STANDARDS IMPLEMENTATION GUIDANCE.
MESSAGE "THE FOLLOWING LIST PROVIDES SPECIFIC GUIDANCE ON"
        "REQUIREMENTS RELATIVE TO THE UNIQUE CHARACTERISTICS"
        "OF YOUR SYSTEM:".
IF SIG=TYPE-B-OR-SOD-FOD THEN
     DISPLAY TEXT(SIG=TYPE-B-OR-SOD-FOD).
```

```
IF SIG=PPS-ALONG THEN
     DISPLAY TEXT(SIG=PPS-ALONE).
IF SIG=RELAX THEN
     DISPLAY TEXT(SIG=RELAX).
IF SIG=FIRMWARE THEN
     DISPLAY TEXT(SIG=FIRMWARE).
IF SIG=PROCESSOR-WAIVER THEN
     DISPLAY TEXT(SIG=PROCESSOR-WAIVER).
IF SIG=SECURE THEN
     DISPLAY TEXT(SIG=SECURE).
IF SIG = INTERSYSTEM THEN
     DISPLAY TEXT(SIG=INTERSYSTEM).
IF SIG = NO-INTERSYSTEM THEN
     DISPLAY TEXT (SIG=NO-INTERSYSTEM).
IF SIG = INTERCOMPUTER THEN
     DISPLAY TEXT(SIG=INTERCOMPUTER)
%
EOF AT LINE 704
```

APPENDIX C - Knowledge Engineering System (KES)

1. INTRODUCTION

Knowledge Engineering System (KES) is both an interactive expert system[1] and a support system for implementing expert systems. As a support system for expert system implementation, KES will parse english-like definitions of a knowledge base into a form suitable for combination with the KES expert system software. The combined result is an operational expert system.

The design of KES reflects certain principles key to its broad applicability . Briefly, the concepts are as follows. First KES is domain independent. That is, it is not tied to any one knowledge area. The software and knowledge base are strictly separated making KES generally applicable to a broad range of problems.

Second, KES provides multiple methods for representing knowledge and for making inferences. Not every problem is effectively addressed with a single inference mechanism. KES provides four techniques - production rules, statistical pattern classification, hypothesize and test, and linear discriminants. Most problems suited to expert systems can be handled by one of these methods.

---

[1] Expert systems are sometimes referred to as knowledge-based expert systems or decision support systems.

Third, KES can be used directly by a domain expert to create an expert system. No programming experience is required. The domain expert creates a knowledge base using a largely non-procedural, english-like language. A minimum of training is required to create executing expert systems.

Fourth, KES is easy to operate. It is designed for use by non-computer professionals. The basic means of interaction is through question and answer. A limited set of commands also is provided to allow user access to the knowledge base outside the standard question and answer protocol. A "help" capability is available for the novice user.

Fifth, KES supports the incorporation of free text in a knowledge base. The term "free text" as used here indicates natural language information that is not processable by computer in the sense of being useable for inference generation. There is a great deal of knowledge in many semi-structured problem solving domains that is appropriate to keep in this form in a knowledge base (definitions, references, and so forth).

C.1 The User's View of KES

The user begins operation of KES by loading the desired subsystem (inference technique) and knowledge base. After the knowledge base is loaded, any KES commands stored as part of the knowledge base's ACTIONS

section are automatically executed. These commands may generate messages to the user or set up goals for the inference mechanism, thereby driving a "mixed-initiative" interactive session.

In general, the user can respond in one of two ways to the multiple-choice questions generated by KES from an active knowledge base. First, he may simply answer the questions as they are asked by entering the appropriate numbers for his selections. This may involve a disjunction (e.g., 1 / 3 / 7) or a conjunction (e.g., 2 & 7) or even a conjunction of disjunctions. In some situations the user may feel inclined to weight the alternatives of a disjunction (e.g., 4<0.6> / 9<0.4>).

The second way a user can respond to a KES-generated question is by ignoring the question and entering a command. When the user enters an unexpected command like this it causes KES to suspend its current processing, set aside the question it just asked, and then enter a control loop in which it asks for commands and executes them. This allows the user to perform a variety of functions such as viewing part of the knowledge base or redirecting the task to which the knowledge base is being applied. The user can resume the suspended command sequence of the knowledge base at will.

C.2 The Domain Expert's View of KES

From the viewpoint of the domain expert, KES is a powerful tool for developing and organizing knowledge-based expert systems. Regardless of the underlying methodology employed to represent and use knowledge, a four step process is involved when building an expert system:

(1) construct a problem-oriented attribute hierarchy;

(2) select an approach to knowledge management;

(3) encode the knowledge base; and

(4) evaluate and certify the resultant expert system.

The first step is conceptually to organize the underlying knowledge in a problem-oriented attribute hierarchy. This structure provides a non-procedural framework around which the knowledge base will be constructed. The term "problem-oriented" emphasizes that each such hierarchy is centered around a specific domain problem.

The second step is to select an appropriate representation format and inference method. This is effectively equivalent to selecting which KES subsystem is best suited to support the expert system.

The third step is to encode the predominantly non-procedural knowledge base for the expert system using

the knowledge representation language of the selected KES subsystem. Knowledge bases are written using a standard text editor and are stored in files. Once written, a knowledge base is submitted by the knowledge base author to the appropriate KES subsystem which parses it for errors much as a compiler examines a high-level language program (e.g., a FORTRAN program). If errors are detected then explanatory error messages are generated. Some typical KES diagnostic error messages are

* * * ERROR: 'HEDACHE' IS AN UNRECOGNIZED NAME * * *

and

* * * ERROR: PRIOR PROBABILITIES DO NOT ADD UP TO 1.0 * * *.

Error messages generally appear immediately under the line of text in the knowledge base where the error is first detected. A knowledge base under development that contains errors can still be used as an expert system, but usually when errors are found by the parser the domain expert will exit KES, correct the knowledge base's errors, and then resubmit it to KES. Once the knowledge base is accepted (i.e., contains no automatically detectable errors), KES begins to execute the commands in its ACTIONS section just as it would were the knowledge base to be activated by a user.

At this point the domain expert passes to the fourth step of the knowledge acquisition process: evaluation and

certification of the expert system that has been built. Evaluation involves testing the accuracy and/or usefulness of the knowledge base's recommendations or prediction in practice. Following testing, a certification history can be kept as part of the knowledge base. This history records such things as who originally wrote the knowledge base, what modifications have subsequently been made to it, and what testing it has undergone.

Appendix D - Implementation Language Analysis

Two critical analyses bearing on the question of SAM/WS software implementation language strategy is the degree of variation of LISP dialects and the feasibility of converting LISP programs to PASCAL. This appendix summarizes the results of these analyses.

D.1 LISP Dialects

Several LISP dialects available through commercial and academic sources were evaluated with respect to the ease with which they could be used to implement expert systems. Table D1 summarizes the various LISP dialects evaluated, the computer architectures they are available on, estimated costs for software when available, type of source, and the overall rating given to the dialect as a vehicle for implementing expert systems.

The properties of LISP which make it a powerful language for implementing expert systems were enumerated in order to perform a meaningful evaluation of each of the LISP dialects examined. These properties included: functions available, utilities supported (i.e., computer, garbage collector, file-handling routines, debug package, etc.), data types available, math routines, documentation, and automatic translation of expert systems written in one dialect to the new dialect. What follows is a summary

of the major problems anticipated with using each LISP when transporting expert systems to different computers.

Overall, there exist several LISP dialects which could be used to implement expert systems. Although most of the problems anticipated are not insurmountable, the use of any particular LISP dialect to implement an expert system must be weighed with the cost in terms of time and manpower needed to overcome these problems. The only problems viewed as insurmountable are the lack of a full complement of data types needed in an expert system implementation language and inadequate memory space. Problems such as the lack of a full complement of functions, functions types, control constructs (especially the PROG feature) are possible to overcome but would require a substantial investment of time and effort to program the needed functions in LISP. The importance of the absence of facilities such as a compiler, or a debugging environment must be viewed as very serious deficiencies.

TABLE D1

NAME	RATING	MACHINE ARCHITECTURE & O.S.	COST	COMMERCIAL/ACADEMIC SOURCE
AlphaLISP	fair	Alpha-Micro System AM-100 AM-100 monitor	NAv*	commercial
IBM LISP/370	fair	IBM 370 VM/CMS, MVS/TSD	$36,000	commercial
InterLISP	excellent	DEC PDP-10 IKA,KI under BBN Tenex DEC 2020 KL-10 under TOPS-20	NAv*	commercial
Cromemco LISP	good	Z-80 based architectures under (DOS or CROMIZ (Cromemco line))	$395	commercial
TLC LISP	good	Z-80 based architectures under CP/M	$150	commercial
DEC PDP-11 LISP (UOM)	not usable	DEC PDP-11 under VOS, DOX	NAv*	academic
MacLISP	good	DEC PDP-10 under ITS, TOPS-10 TENEX (by TOPS-10 emulator); Honeywell MULTICS	NAv*	academic
Franz-LISP	excellent	VAX 11/780 under UNIX	NAv*	academic
LISP Machine LISP	excellent	LMI:CADR, LAMBDA Symbolics: LM-2, 3600	only comes w/machine $45,000-$90,000	commercial
MuLISP/muSTAR 80	good	8080, 8085, Z-80 based architectures under CPM	$200	commercial
Maryland LISP	good	UNIVAC 110/40, UNIVAC 1100/80 under level 36 EXEC 8	NAv*	academic
CDC LISP	-	CL*	CL*	academic
MicrodaSys LISP	-	CL*	CL*	commercial

NAv* - Not available
CL* - Could not Locate

AlphaLISP	:	compiler and garbage collector not supported; poor documentation; all necessary functions not available (i.e., match functions, gensym, C...D, etc.); syntax and semantics of corresponding functions in UNIVAC LISP needed are dissimilar, not easily translated automatically.
IBM LISP/370	:	poor documentation; syntax and semantics of corresponding functions in UNIVAC LISP sometimes dissimilar; uncertain as to whether automatic translation can be accomplished.
InterLISP	:	maximum address space limited to 356K words on some implementations.
Cromemco LISP, TLC LISP	:	no compiler; major control constructs not present (i.e., PROG feature, MAPC); small fixed-sized run time stack; automatic translation not possible.
DEC PDP-11 LISP (UOM)	:	maximum address space 32K words; limited data types; no system commands (i.e., :LISP, :STOP, etc.); limited set of utility functions; no apparent support available.
MACLISP	:	symbols needed for expert systems (i.e., =, -, *, ', etc.) are used as names for predefined functions; poor file I/O capabilities; uncertain support and availability since this is the "father" of newer descendent LISPs.
Franz-LISP	:	symbols needed for expert systems (i.e., +, -, *, ', etc.) are used as names for predefined functions.
LISP Machine LISP	:	no potential problems seen.
MuLISP/MuStar 80	:	poor debugging tools; software support existence questionable; current memory address space limited to 64K words; limited data types; no compiler; control constructs missing (i.e., PROG feature, GO TO); difficult to port expert system to.

Maryland LISP (UOM)	:	KES currently coded in this LISP; could potentially move KES and Maryland LISP to other UNIVAC 1100 series machines.

D.2 LISP to PASCAL Conversion

This section presents the work performed in the evaluation of PASCAL as an implementation language for expert systems. It includes the documentation for key PASCAL data structures, functions, procedures, and parsing schemes needed to implement an expert system. The syntax and semantics of the PASCAL used correspond to WIRTH PASCAL (refer to Jensen, K. and Wirth, N., PASCAL USER MANUAL AND REPORT, Springer-Verlag, N.Y., 1974) except for strings (available in UCSD PASCAL but not "standard" PASCAL).

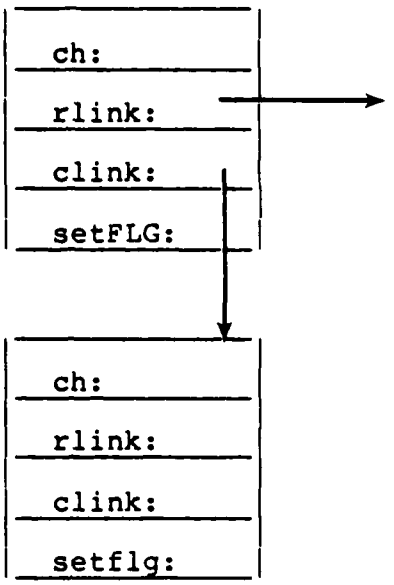
To keep the evaluation time within a reasonable range, a subset of KES.PS referred to as MicroPS was created and used. MicroPS encompasses all of the major concepts embodied in KES.PS and is therefore valid for an evaluation of the type undertaken. A BNF grammar for MicroPS is included in this section.

```
KB → atr-sec % rule-sec % act-sec %
atr-sec → ATTRIBUTES:  atr-decl [.atr-decl]
     atr-decl → name: name [,name]
     name → word [word]
rule-sec → RULES: rule [.rule]
     rule → name IF antecedents THEN consequents
     antecedents → astmt [con astmt]
     astmt → name relation name
     con → & |/
     relation → = | ≠
     consequents → cstmt [& cstmt]
     cstmt → name = name
act-sec → ACTIONS: command [.command]%
     command → STOP|PAUSE|CONTINUE|DISPLAY [string]
               display-option|OBTAIN name|ASKFOR name|
               ASSERT names = name| NEXT next-option|JUSTIFY
     display-option → KB|ATTRIBUTES|RULES|ACTIONS|name|
                      name = name|VALUES|VALUE(name)|
                      RULES (name)
     next-option → CASE|TASK

delimiters: = ≠.,%&/()
```

The following are two possible methods for the parsing of attribute, value, and rule names included within a KES knowledge base

Method 1:

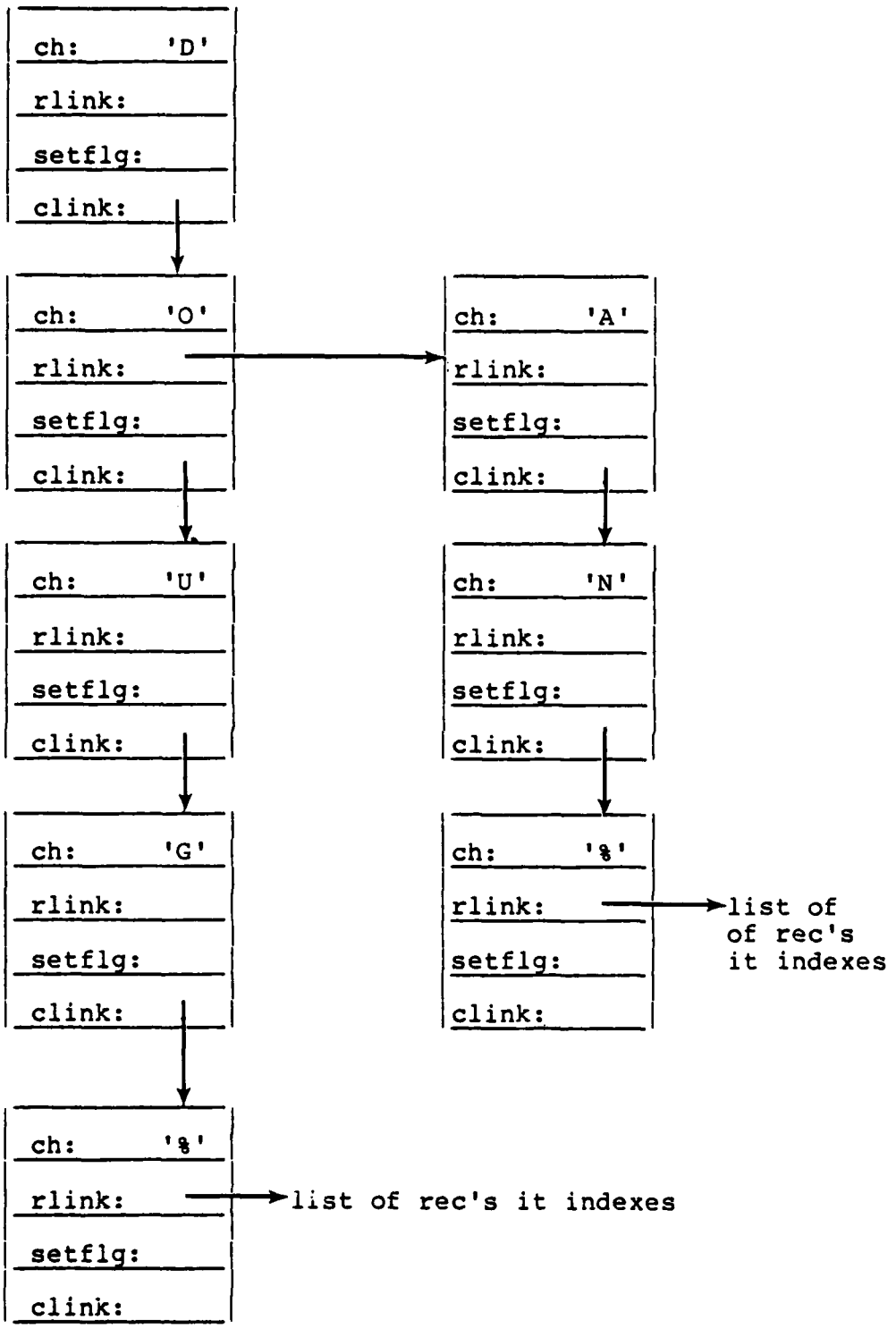
Indexing the names will be via a two-dimensionally linked data structure -



Each node is based on the defined record type data structure namerec -

```
type rlink = (nrec, arec, vrec, rrec);
type namerec = record ch:  character;
                      clink:   namerec;
                      setflg:  BOOLEAN
                      case rlink of
                       nrec:   namerec.
                       arec:   atrec;
                       vvec:   purec;
                       rrec:   rulrec
               end;
```

The rlink will form a list of those unique characters represented in the ch field of namerec which appear in the same character position within different names. Thereby, the "first-row" will be all those unique characters which occupy the first character position in a name. The clink will point to the beginning of another list connected by rlinks which are those characters which occupy the next character position in the name. For example, if only two names DOUG and DAN were parsed respectively, the overall structure would look like (conceptually):


ch: 'D'
rlink:
setflg:
clink:
ch: 'O'
rlink:
setflg:
clink:
ch: 'A'
rlink:
setflg:
clink:
ch: 'U'
rlink:
setflg:
clink:
ch: 'N'
rlink:
setflg:
clink:
ch: 'G'
rlink:
setflg:
clink:
ch: '%'
rlink:
setflg:
clink:
list of
of rec's
it indexes
ch: '%'
rlink:
setflg:
clink:
list of rec's it indexes

A namrec pointed to by a clink with the character '%' in the ch filed marks the end of a name. If a name consisted of more than one word the blank separating the words would be indexed as characters occupying a specific position within the name. If a character within a name encountered at a specific position is not already present in the structure (say the 'G' in DOG, if DOG is the next name being parsed) then a record for that character is dynamically allocated for it and linked to the existing structure appropriately (in this case the G would be in a namerec record pointed to by the rlink of the record containing the character 'V' and the clink for the newly allocated record containing 'G' would point to another newly allocated record which would contain '%' within its ch field denoting the end of a name, 'DOG').

The field SETFLG would be used during set operations (i.e., intersection) for locating names.

Method 2:

The presence of a string data type is assumed in this method. A name would a variable length string up to some maximum length. Again, a record structure would be used with the following three fields: name, rlink, and llink. The records would be dynamically organized into a balanced binary tree with the rlink of a node pointing to the subtree of nodes containing names lexi-

graphically less than the name contrived in its name field and the llink pointing to the subtree of nodes containing names lexigraphically greater than its own.

A separate tree would be maintained for each class of names: attribute, rule, possible values for each attribute. To locate a name, a binary search comparing the strings which compose the desired name would be accomplished. A nice inherent feature in this method is that a pointer to a name contained within a tree structure can be stored in the appropriate record (for instance atrec) so that enforcing the non-duplication of names could be easily realized.

What follows is the PASCAL code for a subset of the data structures, functions, and procedures developed for the evaluation study needed to implement MicroPS.

```
PROGRAM psparse (input, output);

     CONST
        blank = ' '; comma = ','; period = '.';
        equ = '='; colon = ':'; noteqn = '≠';
        percnt = '%'; anddlm = '&'; ordlm = '/';
        paren = '('; rparen = ')'; zero = '0';
        nine = '9'; quote = '"'; dash = '-';


     TYPE
        letter = 'a'..'z';
        digit = '0'..'9';
        tstatus = (und, pos, neg);
        tmisc = (task, cse, kb, atr, rul, act, vals, val);
        object = (ATR, VALUE, CURVAL, RULE, STMT, COM);
        astatus = (Undtrm, unknown, known);
        contype = (aper, slash, zip);
        comname = (display, obtain, stop, pause, continue,
                   assert, next, justify);
```

```
atrec = RECORD name: string;
               status: astatus;
               posval: ↑ pvrec;
               curval: ↑ cvrec;
               rules: ↑ rulrec;
               link: ↑ atrec;
        END;

pvrec = RECORD name: string;
               atr: ↑ atrec;
               link: ↑ pvrec
        END;

cvrec = RECORD id: ↑ pvrec;
               rbd: ↑ rbdrec;
               cf: BOOLEAN;
               link: ↑ cvrec
        END;

rulrec = RECORD name: string;
                status: tstatus;
                antec: ↑ stmrec;
                cnseq: ↑ stmrec;
                link: ↑ rulrec
         END;

rbdrec = RECORD id: ↑ rulrec;
                rbd: ↑ rbdrec;
                type: BOOLEAN
         END;
```

```
stmrec = RECORD con:  ↑stmrec;
                atr:  ↑atrec;
                val:  ↑pvrec;
                link:  ↑stmrec;
                rel:   integer
         END;

comrec = RECORD name:  comname;
                CASE comname OF
                   STOP, PAUSE, CONTINUE, OBTAIN,
                   ASKFOR:  (catr: ↑ atrec);
                   ASSERT, JUSTIFY:  (atr:  ↑ atrec; val:↑ pvrec);
                   DISPLAY:
                      (datr: ↑ atrec; dual: ↑ pvrec;
                      dmsc:  tmisc; strglst:↑ strngrec)
                END;

strngrec = RECORD name:  string;
                  link:↑ strngrec
           END;

VAR
     nxtehr:  char;
     atrlst:  ↑ atrec;
     rullst:  ↑ rulrec;
     comlst: ↑comrec;
     atrptrl:  ↑ atrec;
     rulptrl:  ↑ rulrec;
     comptrl:  ↑ comrec;
     pvptrl:  ↑ pvrec;
     cvptrl:  ↑ cvrec;
     rbdptrl:  ↑ rbdrec;
     stmptrl:  ↑ stmrec;
     strptrl:  ↑ strngrec;

Procedure flagset (P: ↑ glink; VAL:  boolean);
     [used within FUNCTION INTSECT]
     begin
          while P<>NIL do
               begin

                    P ↑ .setflg: = VAL;
                    P:=  P↑ .glink
               end;  [while loop]
     end; (*PROC FLAGSET*)
```

```
Function Intsect (P1; P2: ↑ genrec): ↑ genrec;
        [finds the intersection of two "structures"]
     P3: ↑ genrec;
     result: ↑ genrec;
     begin
          result:=Nil;
          flagset (P1, TRUE);
          while P2<>NIL do
               begin
                  if P2↑.setflg
                     then
                        begin
                             new (P3);
                             P3↑ . isa:=P2↑ . isa;
                             P3↑ . setflg:=NIL;
                             CASE P3↑ . isa OF
                                A: P3↑ . alink:=P2↑ . alink;
                                V: P3↑ . vlink:=P2↑ . vlink;
                                R: P3↑ . rlink:=P2↑ . rlink
                             END;
                             P3↑ . glink:=result;
                             result:=P3
                        END; [if stmnt]
                     P2:=P2↑ . glink
                END; [while loop]
          flagset (P1, FALSE);
          intsect:= result
     END; [function intsect]
```

```
PROCEDURE  OBTAIN (P: ↑ ATREC);
     var ASTATUS: tstatus;

     begin
          if P↑. STATUS=UNDTRM
               then

                    if P↑. RULES=NIL
                      THEN
                           BEGIN
                                ASKFOR (P);
                                P↑. STATUS=KNOWN
                           END
                      ELSE
                           BEGIN

                                ASTATUS:=RULEVAL (P↑. RULES);

                                IF ASTATUS = POS
                                  THEN P↑. STATUS:=KNOWN
                                  ELSE P↑. STATUS:=UNKNOWN
                           END
     END;
```

```
FUNCTION RULEVAL (P: ↑ RULREC): tstatus; [P points to list
                                           of rules]
   VAR RNM, NEXT:  ↑ RULREC;               [RNM points to
                                          "current" rule]
   VAR TALLY     : REAL;                   [TALLY is value of
                                          ant. eval.]
   VAR STATUS    : tstatus;                [NEXT temp. ptr.]

   begin
      RULEVAL:=NEG;
      NEXT:= P;
      WHILE NEXT <>NIL DO
            BEGIN
               STATUS:=NEG;
               IF NEXT↑. STATUS=UND
                  THEN
                     BEGIN
                        NEXT↑. TAL:=ANTEVAL (NEXT↑. ANTEC);
                        TALLY:=NEXT↑. TAL;
                        IF NEXT↑.TAL >0.0
                           THEN
                              BEGIN
                                 RNM:=NEXT;
                                 NEXT↑. STATUS:=pos;
                                 CN (NEXT↑. CNSEQ)
                              END
                           ELSE
                              NEXT↑. STATUS:= neg;
            END
      IF (STATUS=NEG) AND ((NEXT↑. STATUS)=POS)
         THEN RULEVAL=POS;
              NEXT:=NEXT  . LINK
         END [while loop]
END; [proc. RULEVAL]
```

```
FUNCTION ANTEVAL (P: ↑ STMREC): REAL [P points to ant. list]

var TAL: REAL;

    begin
        ANTEVAL:=1.0;
        WHILE (P<> NIL) AND (ANTEVAL >0.0) DO
            BEGIN
                IF P ↑ .CON <>NIL
                    THEN TAL:=EVALOR(P)
                    ELSE TAL:= RELATION (P ↑ . ATR ↑ .CURVAL,
                                P ↑ .VAL, P ↑ . REL);
                IF ANTEVAL > TAL THEN ANTEVAL:= TAL;
                P:= P ↑ . LINK;
    END
END;

FUNCTION HSVAL (P: ↑ ATREC, Q: ↑ PVREC, REL: INTEGER): BOOLEAN;
    var  ANSWER: REAL;

    begin
        ANSWER:=RELATION (P↑.CURVAL, Q, REL);
        if ANSWER= 1.0
            THEN HSVAL:= true
            ELSE HSVAL:= false;
END;

FUNCTION EVALOR (P: ↑ STMREC): REAL;
     var TAL: REAL;

     begin
         EVALOR:= -1.0;
         WHILE (P<>NIL) AND (EVALOR<1.0) DO
              BEGIN
                  TAL:= RELATION (P ↑ . ATR ↑ . CURVAL, P ↑ . VAL,
                       P↑. REL);
                  IF EVALOR <TAL  THEN EVALOR:= TAL;
                  P:= P ↑ . CON
      END
END;
```

```
FUNCTION RELATION (P: ↑ CVREC, Q: ↑ PVREC, REL: INTEGER):REAL;
    var value, curvals: string;
    var nfound: Boolean;

    begin
        relation:=-1.0;
        value:=Q↑ . NAME;
        nfound:= true;
        While (P<>NIL) AND (NFOUND) DO
            BEGIN
                CURVALS:= P↑ . NAME;
                IF CURVALS = VALUE
                    THEN
                        BEGIN
                            if P↑ . CF = TRUE
                                then relation:= 1.0
                                else relation:=-1.0;
                            nfound:= false
                        END
                    else
                        P:= P↑ . LINK
        END
    if REL=2 then relation:= -relation
END; [FUNCTION RELATION]
```

```
PROCEDURE CN (P: ↑ STMREC); [P points to start of conseq. list]
    var type, nfound: BOOLEAN;
    var Q, Q3: ↑ CVREC;
    var Q2: ↑ rbdrec;

    begin
[L1]     while P<>NIL DO
             BEGIN
                IF P↑ . REL=2      [if ≠ then RD else RB]
                    THEN type:= false
                    else type:= true;
             Q:= P↑ . ATR↑ . CURVAL    [find if CV already present]
             nfound:= true;
             While (Q<>NIL) AND (NFOUND) DO
                if Q↑ . ID↑ . NAME = P↑ . VAL↑ . NAME
                    THEN
                        BEGIN
                           nfound:= false;
                           new (rbdptrl)'
                           Q2:= Q↑ . rbd;
                           While (Q2↑.rbd<>NIL) DO
                               Q2:= Q2↑ . rbd;
                           Q2↑ . rbd:= rbdptrl;
                           IF (Q↑ . CF = TRUE) AND (TYPE = FALSE)
                               THEN Q↑ . CF = FALSE
                        END
                    ELSE
                        Q:= Q↑ . link;
                if nfound     [value not in list of CV's)
                    THEN
                        BEGIN
                            new (cvptrl);
                            new (rbdptrl);
                            cvptrl ↑ . rbd:= rbdptrl;
                            cvptrl↑ . cf;= type;
                            Q3:= P↑ . ATR↑ .CURVAL;
                            While Q3↑ . link <>NIL DO
                                  [find last CV]
                                  Q3: = Q3↑ . link;
                                  Q3↑ . link: = CVPTR1;
                            cvptrl.id:= P↑ . VAL
                         END;

                rbdptrl↑ . id:= RNM;
                rbdptrl↑ . type:= type;
                [get next cnseq]
                P:= P↑ . link
        END [L1 loop]
END;        [proc CN)
```


ILMED
8